Volume 19 Number 1

ACTA CYBERNETICA

Szeged, 2009

ACTA CYBERNETICA

Information for authors. Acta Cybernetica publishes only original papers in the field of Computer Science. Manuscripts must be written in good English. Contributions are accepted for review with the understanding that the same work has not been published elsewhere. Papers previously published in conference proceedings, digests, preprints are eligible for consideration provided that the author informs the Editor at the time of submission and that the papers have undergone substantial revision. If authors have used their own previously published material as a basis for a new submission, they are required to cite the previous work(s) and very clearly indicate how the new submission offers substantively novel or different contributions beyond those of the previously published work(s). Each submission is peer-reviewed by at least two referees. The length of the review process depends on many factors such as the availability of an Editor and the time it takes to locate qualified reviewers. Usually, a review process takes 6 months to be completed. There are no page charges. Fifty reprints are supplied for each article published.

Manuscript Formatting Requirements. All submissions must include a title page with the following elements:

- title of the paper
- author name(s) and affiliation
- name, address and email of the corresponding author
- An abstract clearly stating the nature and significance of the paper. Abstracts must not include mathematical expressions or bibliographic references.

References should appear in a separate bibliography at the end of the paper, with items in alphabetical order referred to by numerals in square brackets. Please prepare your submission as one single PostScript or PDF file including all elements of the manuscript (title page, main text, illustrations, bibliography, etc.). Manuscripts must be submitted by email as a single attachment to either the most competent Editor, the Managing Editor, or the Editor-in-Chief. In addition, your email has to contain the information appearing on the title page as plain ASCII text. When your paper is accepted for publication, you will be asked to send the complete electronic version of your manuscript to the Managing Editor. For technical reasons we can only accept files in LaTeX format.

Subscription Information. Acta Cybernetica is published by the Institute of Informatics, University of Szeged, Hungary. Each volume consists of four issues, two issues are published in a calendar year. Subscription rates for one issue are as follows: 5000 Ft within Hungary, €40 outside Hungary. Special rates for distributors and bulk orders are available upon request from the publisher. Printed issues are delivered by surface mail in Europe, and by air mail to overseas countries. Claims for missing issues are accepted within six months from the publication date. Please address all requests to:

Acta Cybernetica, Institute of Informatics, University of Szeged
P.O. Box 652, H-6701 Szeged, Hungary
Tel: +36 62 546 396, Fax: +36 62 546 397, Email: acta@inf.u-szeged.hu

Web access. The above informations along with the contents of past issues are available at the Acta Cybernetica homepage http://www.inf.u-szeged.hu/actacybernetica/ .

Bálint Dömölki
IQSOFT
Budapest, Hungary
domolki@iqsoft.hu

Zoltán Ésik
Department of Foundations of
Computer Science
University of Szeged
Szeged, Hungary
ze@inf.u-szeged.hu

Zoltán Fülöp
Department of Foundations of
Computer Science
University of Szeged
Szeged, Hungary
fulop@inf.u-szeged.hu

Ferenc Gécseg
Department of Computer Algorithms
and Artificial Intelligence
University of Szeged
Szeged, Hungary
gecseg@inf.u-szeged.hu

Jozef Gruska
Institute of Informatics/Mathematics
Slovak Academy of Science
Bratislava, Slovakia
gruska@savba.sk

Tibor Gyimóthy
Department of Software Engineering
University of Szeged
Szeged, Hungary
gyimothy@inf.u-szeged.hu

Helmut Jürgensen
Department of Computer Science
Middlesex College
The University of Western Ontario
London, Canada
helmut@csd.uwo.ca

Alice Kelemenová
Institute of Computer Science
Silesian University at Opava
Opava, Czech Republic
Alica.Kelemenova@fpf.slu.cz

László Lovász
Department of Computer Science
Eötvös Loránd University
Budapest, Hungary
lovasz@cs.elte.hu

Gheorghe Păun
Institute of Mathematics of the
Romanian Academy
Bucharest, Romania
George.Paun@imar.ro

András Prékopa
Department of Operations Research
Eötvös Loránd University
Budapest, Hungary
prekopa@cs.elte.hu

Arto Salomaa
Department of Mathematics
University of Turku
Turku, Finland
asalomaa@utu.fi

László Varga
Department of Software Technology
and Methodology
Eötvös Loránd University
Budapest, Hungary
varga@ludens.elte.hu

Heiko Vogler
Department of Computer Science
Dresden University of Technology
Dresden, Germany
vogler@inf.tu-dresden.de

Gerhard J. Woeginger
Department of Mathematics and
Computer Science
Eindhoven University of Technology
Eindhoven, The Netherlands
gwoegi@win.tue.nl

Conference of PhD Students in Computer Science

Guest Editor:

Kálmán Palágyi

Department of Image Processing and Comuter Graphics
University of Szeged
Szeged, Hungary
palagyi@inf.u-szeged.hu

Preface

The sixth Conference for PhD Students in Computer Science (CSCS) was organized by the Department of Computer Science of the University of Szeged (SZTE) and held in Szeged, Hungary from July 2-5, 2008. The members of the Scientific Committee were the following representants of the Hungarian doctoral schools in computer science: András Benczúr (ELTE), Hasszan Charaf (BME), Tibor Csendes (SZTE), János Csirik (SZTE), József Dombi (SZTE), Zoltán Ésik (SZTE), Zoltán Fülöp (SZTE), Ferenc Gécseg (Chair, SZTE), Tibor Gyimóthy (SZTE), Zoltán Horváth (ELTE), Zoltán Kató (SZTE), Zoltán Kása (Sapientia EMTE) János Kormos (DE), László Kozma (ELTE), Eörs Máté (SZTE), Attila Pethő (DE) András Recski (BME), Endre Selényi (BME), and Tamás Szirányi (SZTAKI). The members of the Organizing Committee were Balázs Bánhelyi, Tamás Gergely, István Matijevics, and Kálmán Palágyi (chair).

There were more than 60 participants and 39 talks in several fields of computer science and its applications. Beyond the Hungarian PhD schools in computer science, 5 other European countries were represented. The talks were going in sections in artificial intelligence, automata and formal languages, computer networks, database theory, discrete mathematics, fuzzy decision support systems, information systems, optimization, image processing, and software engineering. The talks of the students were completed by four plenary talks of leading scientists: Tamás Horváth (Univ. Bonn and Fraunhofer IAIS), Gábor Ivanyos (Computer and Automation Research Institute, Hungarian Academy of Sciences), Márk Jelasity (Univ. Szeged), and Zora Konjovic (Univ. Novi Sad).

Three scientific journals, viz. Periodica Polytechnica (Budapest), Publicationes Mathematicae (Debrecen) and Acta Cybernetica (Szeged) offered students to publish the paper version of their presentations after a selection and review process. Altogether 20 manuscripts were submitted for publication. The present special issue of Acta Cybernetica contains 8 such papers.

The full program of the conference, the collection of the abstracts and further information can be found at `http://www.inf.u-szeged.hu/~cscs`.

On the basis of our repeated positive experiences, the conference will be organized in the future, too, hopefully with more foreign participants. According to the present plans, the next meeting will be held in July 2010 in Szeged.

Kálmán Palágyi
Guest Editor

Acta Cybernetica **19** (2009) 5–15.

Backprojection Reconstruction Algorithm Using Order Statistic Filters In Breast Tomosynthesis

László Csernetics

Abstract

Breast cancer is the most common cancer type and one of the leading cause of death among women. It has been recognized over the years that preventing the disease is the most powerful weapon, and the implementation of screening mammography has had significantly reduced the death rate. However, it is also proven that conventional mammography does not detect approximately 30% of breast cancers. Inventing new imaging technologies for the earlier detection of breast cancer is vital and is in the center of many ongoing studies. There are several new techniques using different imaging modalities that are under investigation. The most promising is the breast tomosynthesis, an advanced x-ray application that addresses the problem of structure superimposition, one of the major deficiencies of 2D mammography, by reconstructing a range of slices providing additional 3-dimensional information of the breasts.

Our goal is to investigate and develop reconstruction algorithms that fit into the new mathematical model of tomosynthesis used in mammography. In this paper we show a backprojection reconstruction technique that is especially well-suited for the problem in question. This algorithm is capable to produce contrast-enhanced slices of the breast by taking only the projections that most probably hold the "important" information of the targeted lesions, ignoring part of the projections. This statistical approach also offers a good noise management performance, as a fortunate side-effect. After discussing the algorithm we publish the results of the comparison of this technique with other popular methods of the algorithm-family. We also look out the strict boundaries of the work done suggesting improvements of the reconstruction algorithm.

Keywords: breast tomosynthesis, mammography, breast cancer, reconstruction algorithm

1 Introduction

Today, breast cancer is the most common type of cancer among women, about 1.3 million new cases will be diagnosed in the near future annually worldwide, and about 465,000 will die from this disease, according to the American Cancer Society.

With other words, in the future one of every 8 women is expected to be diagnosed with breast cancer at some point in her lifetime. On the other hand, the death rate of this type of cancer is steadily dropping since the 90's, due to the successful breast imaging techniques introduced and used in the practice of screening exams, especially in the developed countries. Although the death rate has been halved over the years, it has been proven that the commonly used mammographic techniques are still imperfect tools. In fact, around 30% of breast cancers are missed in the first year of presence, because of the lack of sensitivity and specificity of the conventional mammography ([6, 8, 9]).

The limitations of mammography are originated from the 2D characteristics of the technique. The healthy (functional- and fatty-) tissues and the potential abnormalities are superimposed on the 2D image, making it difficult to detect and classify the different structures of the breast. The overlapped normal structures could imitate abnormalities, resulting in additional, but unnecessary examinations of the patient. In other cases, the abnormal structures could be surrounded and hidden within the normal tissues of the breast, causing false-negative detections, and increasing the risk of developing breast cancer (refer to [1] for more detailed comparison of mammographic techniques).

Breast tomosynthesis, a new 3D mammographic technique that is still under development, is targeting exactly this deficiency of conventional mammography. By reconstructing a range of 2-dimensional horizontal slices, it provides much more visual information of the breast. Moreover, by using new reconstruction algorithms that are fine-tuned for breast tomosynthesis, the contrast of the structures could be improved, helping the physicians to establish more precise diagnosis. This is also the subject of our research, and we are going to show such an algorithm in this paper.

2 Digital breast tomosynthesis

Digital breast tomosynthesis is a technique on the way from the conventional mammography to the computerized tomography ([10, 11]). The procedure of image acquisition and also the device itself are very similar to the currently applied technology, as it has been developed from 2D mammography. Therefore, the patients are already familiar with it, what is more, this new technique is even more comfortable. It preserves the advantages of mammography, most importantly the low radiation dose used during an examination. The only difference becomes evident with the quality of the produced images, as well as the efficiency of the method. It is clear, that this technique is about to replace today's imaging modalities of screening examinations.

Tomosynthesis is an image processing technique for generating 2-dimensional slices of the imaged organ, while all the information that is needed for such a reconstruction is not necessarily available. Usually, only a narrow angular range of image acquisition is given, and only a limited number of projections can be taken. The tomographic techniques commonly used in medical imaging are not

facing such constraints, and in those cases the complete 3D data of the imaged volume can be reconstructed. This is not possible in the case of tomosynthesis, where only a series of slices can be reconstructed, containing an additional depth information, that becomes clearly visible when the reader of the mammogram loops through the slices. This is what tomosynthesis really offers for the physicians, it virtually eliminates the structure superimposition presented on the conventional mammograms.

3 Projection geometry

The story of breast tomosynthesis is quite new, the first studies have been published in the 90's, and the system has still not been introduced into the practical usage. In spite of this fact, there are already several independent research groups, including the pioneers of the field from the Massachusetts General Hospital (MGH), and the research team of the Duke University Medical Center, working for the same goal. One of the main differences we can find between the actually ongoing studies is the geometrical model implemented with the system.

Generally, we can speak about two main models whose basic concepts are the same, and they differ only in a few minor geometrical aspects, including the number of projections taken and the angular range of x-ray tube rotation. In our work, we are referring the model used by the group at the MGH, also published in [2, 7, 8]. Its schematic representation can be seen in Figure 1.

In tomosynthesis, the x-ray detector is stationary, while the x-ray tube is rotating during the image acquisition. Before an examination starts, the breast has to be fixated with the compression paddles of the device, located above the detector. This step is needed only for the immobilization of the breast, no real compression like in conventional mammography is applied (this is why this method is much more comfortable). The imaging process itself takes only 7 seconds, and 11 low radiation dose images are taken. During this time, only the x-ray tube is moving on an arc of 50°, starting from the tilted position of +25°, making 5° angle increments for every image. The distance between the x-ray tube and the detector is 66cm, where the axis of rotation is at 44.3cm from the source, and at 21.7cm from the detector. The sixth image is taken from the position where the electron rays are hitting the detector at a right angle (like the CC view of the conventional mammography). The total radiation dose of such an examination is about 1.5 times more than in the 2D mammography.

Using the projections produced by the above mentioned way, it is possible to reconstruct the 3D dataset of the imaged volume. However, the limited input of the model states a difficult image reconstruction problem. It is obvious, that the commonly used basic reconstruction algorithms will not provide the desired results, because of the high error ratio indicated by the very few number of projections, as well as the narrow angular range. An algorithm using some kind of *a-priori* knowledge, that is capable to deal with the error and noise presented in the input images has to be employed. As an additional aim, it would be nice to extract the

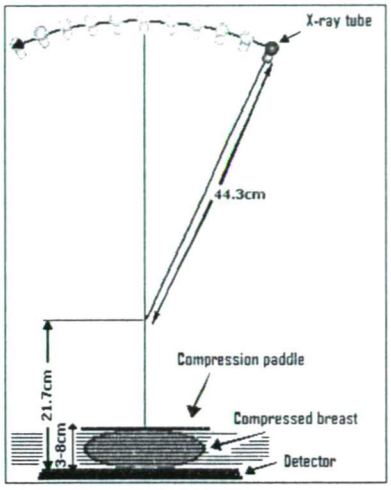


Figure 1: The schematic representation of the used projection geometry. The x-ray tube is in the starting position.

targeted structures on the results. In the second part of the paper, we are going to show an algorithm that is especially well-suited to breast tomosynthesis, and also able to produce contrast-enhanced reconstructed slices of the imaged breast.

4 Order statistic based reconstruction algorithm

Generally speaking, reconstructing an object from its projections is the inverse transformation of the projection generation process itself. Reconstruction algorithms can be divided into two main groups, regarding the way of approaching the result (more about reconstruction algorithms can be found in [4]). The MLEM (Maximum-Likelihood Expectation Maximization), as an iterative algorithm, represents one of the most successful methods of doing the reconstruction in breast tomosynthesis, however, it is very complex, and the reconstruction of a high quality image set usually takes several hours. On the other hand, the non-iterative algorithms are much more simple, since they are simply projecting back the projections into the volume to be reconstructed. This is why these methods are the most widely accepted techniques in the field. In our work, we are investigating a variant of the conventional filtered backprojection algorithm, that has already been employed in the tomosynthesis studies for angiography, decades ago. Beside simply adopting the technique onto the field of breast tomosynthesis, we are going to prove its validness for the problem in question. We have to emphasize that this is a heuristic approach. At some points the formulation is only a crude approximation, rather than a mathematically precise derivation of the algorithm.

Before going into the details of the algorithm, let us take a closer look at the

already mentioned transformations. The projection, known also as the Radon-transform, has the following form:

$$p^{\phi}(s_1, s_2) = \int_{-\infty}^{\infty} f^{\phi}(s_1, t, s_2)dt = [Rf^{\phi}](s_1, s_2), \tag{1}$$

where p^{ϕ} denotes the 2D projection taken under the angle ϕ, and f^{ϕ} denotes the 3D object in the rotated coordinate system of the device. The backprojection operator can be written up as the formal adjoint of the Radon-transform:

$$[R^*p](x, y, z) = \tilde{f}(x, y, z) = \int_{-0}^{0} p^{\phi}(x\cos\phi + y\sin\phi, z)d\phi. \tag{2}$$

However, this is not the inversion of the Radon-transform, it gives a blurred recovery of the original image. In fact, applying the backprojector operator on R results in the convolution of the original image function with a blurring function hl:

$$[R^*Rf] = \tilde{f} = (f * hl). \tag{3}$$

The image could be fully recovered only by applying the appropriate inverse filtering on $\tilde{f}$.

We can notice, that in our case the third dimension is not affecting the calculations, since we have a tilted geometry, with only one degree of freedom (tilting around the z axis). After a normalization with the number of projections, we get the appropriate averages of the backprojected values in every point of the volume. Discretizing the formula in (2) will finally lead us to the basic equation of the simple backprojection algorithm:

$$V(X) = \tilde{f}(x, y, z) = \frac{1}{k} \cdot \sum_{n=1}^{k} P_n(\bar{X}), \tag{4}$$

where $V(X)$ denotes the value of the voxel belonging to the position determined by (x, y, z), k denotes the number of projections, and $P_n(\bar{X})$ denotes the pixel value of the nth projection at the position calculated from the above mentioned coordinate triplet.

This simple backprojection algorithm is mathematically clear, straightforward to implement, and has low computational costs. However, since the projection values are equally distributed on the ray-path belonging to a certain projection position, and the reconstructed values are simply the averages of the backprojected values, the result of this method is not satisfactory. Applying filters in the frequency domain of the projections is a very common way of improving the quality of the reconstruction. Still, in our case this is not enough to achieve the best possible result, because of the limited input data. To make this method even more suitable for breast tomosynthesis, the basic conception of the algorithm has to be modified as well.

In the middle of 80's, researchers from Hamburg have been introduced a new non-linear method, as they called, the extreme-value reconstruction ([3, 5]). The

main idea was to replace the averaging operator used in (4) with a minimum-type operator:

$$V^*(X) = \min_{n=1}^{k}\{P_n(\bar{X})\}. \tag{5}$$

This way the reconstructed voxels were holding the minimum of the backprojected values instead of their average, which is convenient in the case of angiography, since the expected values are approximately 0 everywhere, except the vessels, due to the injected contrast material (Digital Subtraction Angiography).

When we examine a few mammograms, we can see, that the situation is very similar to the angiography. The fatty tissues are appearing as very dark areas, the functional and other healthy tissues are brighter, while the abnormal structures are very bright, or white regions. Since we are primarily interested in the correct reconstruction of the abnormal structures potentially presented in the breast, after an appropriate preprocessing of the projections, we can apply the above mentioned extreme-value reconstruction. On the other hand, it is also obvious that there is a relation between the number of projections used in the reconstruction of a voxel and the noise-characteristics of the reconstruction algorithm itself. While averaging the projections in a reconstruction step has a noise reducing effect, using only one projection will end up in a much more noisy result, because of the extreme-value selection. Thus, combining the two operators could lead us to a more optimal result. As a first approach, we can construct an averaging operator that averages all, but the K largest projections (to preserve the advantages of the minimum-type operator). But, ignoring also the minimal (generally, the L smallest) value could improve the noise-management of the algorithm. This way we finally end up with the so called order statistic filter (OS-filter, or L-filter):

$$V^{**}(X) = \frac{1}{(k-L-K)} \cdot \sum_{n=L+1}^{k-K} P_n(\bar{X}), \tag{6}$$

where the projections are ordered by their value at the position $\bar{X}$ every time a voxel is being reconstructed:

$$P_{min}(\bar{X}) \leq \cdots \leq \underbrace{P_{L+1}(\bar{X}) \leq \cdots \leq P_{k-K}(\bar{X})} \leq \cdots \leq P_{max}(\bar{X}). \tag{7}$$

In the case, when there are more projections with the same value in the series above, the randomly permutated order of these values should be used in order to avoid some projections to be preferred to others. We have to emphasize, that we are going to employ a minimum-type operator, so the value of L should be chosen to be small, while the K should be larger. Regarding to the observations done so far, we suggest the selection of $\{L = 2, K = 4\}$.

Note the relationship between the equations covered above. We can write up these operators with one common formula:

$$\widetilde{V}(X) = \frac{\sum_{n=1}^{k} \omega_n P_n(\bar{X})}{\sum_{n=1}^{k} \omega_n}, \tag{8}$$

where the selection of the ω_n weights determines the type of the operator described with the formula (e.g. $\omega_n = 1$ for every n is the averaging operator). The appropriate selection of the weights depends on the *a-priori* knowledge associated with the problem. Particularly, the filter introduced above is defined by the following weight selection strategy:

$$\omega_n = \begin{cases} 1 & \text{if } P_{L+1}(\bar{X}) \leq P_n(\bar{X}) \leq P_{k-K}(\bar{X}) \\ 0 & \text{otherwise} \end{cases}, \tag{9}$$

where the indexing of the projections is in accordance with (7).

From this point we are going to discuss the reconstruction algorithm. Before starting with the first step, a pre-processing of the input projection images is needed. The purpose of this step is to correct the errors caused by the ray scattering effect during the image acquisition, as well as to normalize the projections against the geometrical distortion presented in the images caused by the varying ray-path lengths within the imaged breast under the varying angles of incidence. After this correction, every pixel of the projections will hold the average attenuation value measured on the corresponding ray-path. We can also make a background subtraction following the idea from digital subtraction angiography, assuming that the fatty tissues are appearing as homogeneous, dark areas on the images. It is important, however, to perform this kind of correction globally, to preserve the images normalized. Since the algorithm is based on projection ordering, it is very sensitive to intensity-level differences presented between the projections. This is also the reason for leaving the projections unfiltered for the reconstruction.

In the first step of the algorithm we are doing a reconstruction using the operator in (6) as the basis of the backprojection algorithm. As it was expected, the reconstructed horizontal slices show reduced error rate caused by the so called out-of-plane artifacts, the side-effect of the averaging operator. Also, the noise-management has been improved comparing to the extreme-value reconstruction. But, in spite of the seemingly good results, our algorithm is completely incorrect in the mathematical point of view. When we analyze the result of the reconstruction, it becomes obvious that the re-projection consistency constraint won't be satisfied. This criterion states against a reconstruction algorithm to produce a result that is consistent with the projection data, i.e. if a second projection is applied on the reconstructed data, the very same projections should be produced as in the case of the original object. Mathematically, since the projection values are representing the average attenuation values along the corresponding ray-paths, this means the following:

$$P_n(x) = \frac{1}{N} \cdot \sum_{i=1}^{N} R_i(\bar{x}), \tag{10}$$

where $P_n(x)$ is the value of nth projection at the detector-position x, N is the number of the reconstructed slices, and $R_i(\bar{x})$ is the value of the pixel $\bar{x}$ on ith slice. Of course, this would be satisfied in the case when the projection values are distributed equally on all of the reconstructed slices (like in the case of the

averaging operator), but not in our case, when part of the projections is simply ignored during the reconstruction of each of the voxels.

As the second step of the algorithm, we are going to modify the original projections in such a way, that after doing a second order-statistic based reconstruction the re-projection consistency constraint will become satisfied. Let's first write up the equation in (10) in the following form (from this point we will denote the ith slice of the first and the final reconstruction with S_i and R_i, respectively):

$$P_u(x) = \frac{1}{N} \cdot \sum_{i \in \kappa} R_i(\bar{x}) + \frac{1}{N} \cdot \sum_{i \notin \kappa} R_i(\bar{x}), \tag{11}$$

where $i = 1, \ldots, N$, and $\kappa \subseteq \{1, \ldots, N\}$ is the set of slices where the $P_u(x)$ value has no contribution in the reconstruction of the pixel $\bar{x}$. The elements of the κ set is being collected during the first reconstruction step, for each pixel on every projection. Since the reconstructed values found at $\bar{x}$ on these slices are not affected by the projection value $P_u(x)$, we can accept the result of the first reconstruction at these positions. However, since the modification of the other projections could have an influence on these values, we are accepting these values only as the approximations of the final reconstruction values. Thus:

$$\sum_{i \in \kappa} R_i(\bar{x}) \approx \sum_{i \in \kappa} S_i(\bar{x}). \tag{12}$$

Rewriting (11) we get:

$$\sum_{i \notin \kappa} R_i(\bar{x}) \approx N \cdot \left(P_u(x) - \frac{1}{N} \cdot \sum_{i \in \kappa} S_i(\bar{x}) \right), \tag{13}$$

the approximate sum of the reconstructed values to which also the x pixel from the projection P_u had contribution during the first reconstruction. Since it is assumed that, after the pre-processing step, every projection value is representing the average attenuation, and the number of slices receiving contributions from $P_u(x)$ is N minus the cardinality of κ, we have to modify our projections in the following way:

$$\widetilde{P}_u(x) = \frac{N}{N - |\kappa|} \cdot \left(P_u(x) - \frac{1}{N} \cdot \sum_{i \in \kappa} S_i(\bar{x}) \right). \tag{14}$$

Now, that we have these enhanced projections, in the third step of the algorithm we are doing a final reconstruction of the horizontal slices. As in the previous case, the technique preserves the advantageous behavior of both the minimum-type and the averaging operators. Furthermore, the result is now mathematically correct, or, at least a mathematically correct approximation, regarding to the re-projection consistency constraint. As a fortunate side-effect of the projection modification, the contrast of the objects has been improved in the result, which was our additional criterion against the algorithm. This contrast enhancement could be explained by

the characteristics of the κ set. Note, that a projection value and the cardinality of the corresponding κ set are linearly related, since the higher values are more probably ignored during the reconstruction. At the same time, it is obvious from (14) that the projections ignored in most of the cases will be modified in the highest degree. This way, in the final result, the smaller, but higher intensity objects will have an increased contrast. As we know, this is very well suited to the breast tomosynthesis, since we are interested in discovering of such kind of structures.

5 Experimental results

In our experiments we compared the order-statistic based backprojection algorithm with the algorithms employing the conventional averaging and the extreme-value operator. We ran the tests for ideal and noisy projections as well.

The experiments have been performed on an Intel P4 2.4GHz machine, with 1GB RAM memory, and Windows XP operation system on it. We have generated breast phantoms containing several objects of different sizes and placements, that are responsible for simulating the inner structure of the breast. 27 of them are bigger spheres filling out the volume, and representing the healthy mass of a breast. The other 7 spheres are smaller, but having more concentrated mass, thus, higher attenuation coefficient, and imitating different kind of abnormalities. 5 of these objects are very small, simulating calcium deposits in the breast. These calcifications could be the first signs of a malign abnormality, so it is important to find them as soon as possible. The other 2 bigger objects are imitating already developed abnormal structures embedded within the healthy tissues. The parameters of the algorithm have been set to $\{L = 2, K = 4\}$, and 21 horizontal slices of size 512x300 have been reconstructed with each algorithm, separately. The results can be seen in Figure 2.

Regarding to the time complexity of these algorithms, they are performing very similarly, as it could be expected. The average time needed to finish a complete reconstruction (including the two runs of the backprojection algorithm, as well as the projection modification step) was 153.15 seconds. This is acceptable comparing to the time consumption of an iterative algorithm.

Looking into the reconstructed images it can be seen that the algorithm using the averaging operator produced fairly the worst results. Although, the noise-management was quite good, the objects on the slices are low-contrasted, thus, this method is not convenient to detect the structures in question. On the other hand, the minimum operator was able to restore the shapes nicely, but in the noisy case it performed weakly. Eventually, the algorithm presented in this paper produced the best results, tolerating the noise and increasing the contrast of the objects at the same time.

Of course, regardless to the good results given, the algorithm has to be developed further in the future. We are supposing, that an iterative projection enhancing technique could improve the results, since the applied approximation during the derivation is converging toward satisfying the re-projection consistency constraint.

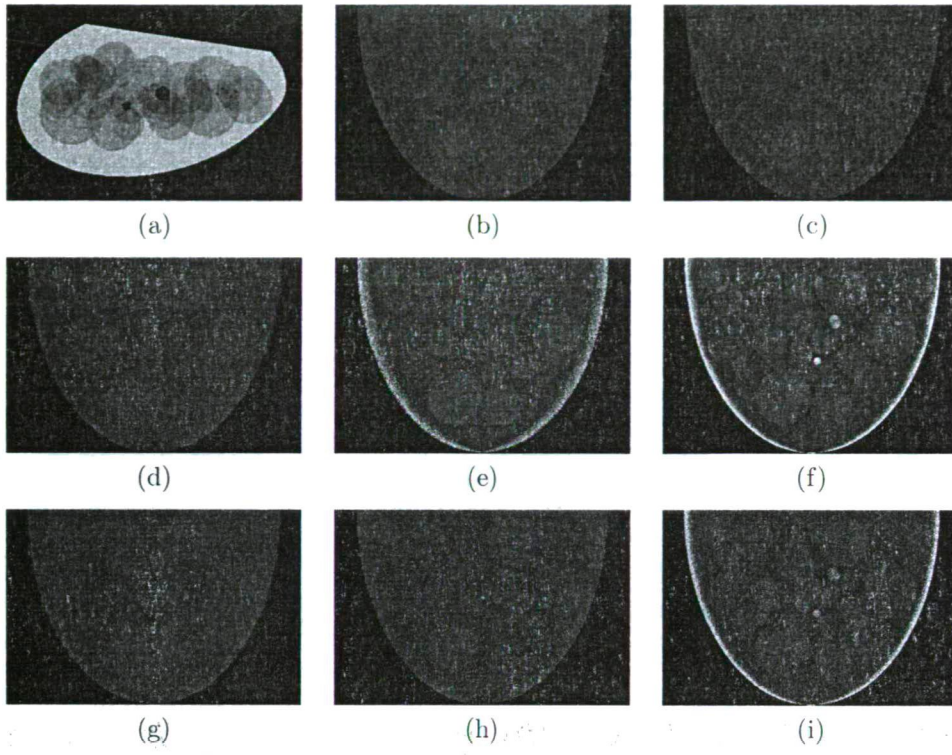


Figure 2: The top row presents the 3D breast phantom containing the 7 abnormalities with darker colors (a), the ideal (b), and the noisy 6th projection (c). The two rows below contains the central (11th) reconstructed slice in the ideal (d-f) and noisy (g-i) case: using the averaging operator (d, g), the minimum operator (e, h), and the order-statistic based operator (f, i), respectively. As it is clearly visible, this latter produced the most contrasted result, especially in the point of view of the targeted objects.

Also, more realistic breast phantoms, or even real data as input of the algorithm should be used in the experiments, to get more accurate results. Finally, the reconstructed slices themselves could be also improved by employing image processing techniques for image refinement.

Acknowledgements

The author would like to express thanks to his mentor, Dr. Attila Kuba, who supported this work with a great deal of help in research. Without his great experience and knowledge these results could not have been born. The author also received a considerable deal of support and inspiration to prepare this paper and keep on

studying the research topic from his supervisor, Dr. Kálmán Palágyi. Finally, the author is thankful for the valuable comments to the reviewers of the paper.

References

[1] American Cancer Society: Mammograms and other breast imaging procedures. Last revised: 09/26/2008. Accessed August 2008.

[2] Csernetics, L.: Melltomoszintézis: 3D Mammográfia. Proc. 6th Conference of Hungarian Association for Image Processing and Pattern Recognition (2007) 9–19

[3] Haaker, P., Klotz, E., Koppe, R., Linde, R., Möller, H.: A new digital tomosynthesis method with less artifacts for angiography. Med. Phys. **12(4)** (1985) 431–436

[4] Herman, G.T.: Image reconstruction from projections. The fundamentals of computerized tomography. Academic Press, NY (1980) 1–40, 55–65, 90–147

[5] Klotz, E., Haaker, P., Koppe, R., Linde, R.: Three-dimensional angiography with digital flashing tomosynthesis. Adv. Imag. Proc. SPIE Proc. **804** (1987) 306–313

[6] Pisano, E.D., Gatsonis, C., Hendrick, E., Yaffe, M., Baum, J.K., Acharyya, S., Conant, E.F., Fajardo, L.L., Bassett, L., D'Orsi, C., Jong, R., Rebner, M.; Digital Mammographic Imaging Screening Trial (DMIST) Investigators Group: Diagnostic performance of digital versus film mammography for breast-cancer screening. N Engl J Med **353(17)** (2005) 1773–1783

[7] Rafferty, E.A.: Advances in imaging: Breast tomosynthesis. ASCO, Virtual Meeting (2004)

[8] Rafferty, E.A.: Breast tomosynthesis: Three-dimensional mammography. ASCO, 40th Annual Meeting (2004) 45–47

[9] Rafferty, E.A.: Tomosynthesis: New weapon in breast cancer fight. Decisions in Imaging Economics **17(4)** (2004)

[10] Smith, A.P.: Contrast enhanced breast tomosynthesis: A promising tool for identifying breast cancer. Hospital Imaging & Radiology Europe **1(3)** (2006)

[11] Wu, T., Rafferty, E.A., Kopans, D.B., Moore, R.H.: Contrast-enhanced breast tomosynthesis. DBT for better breast cancer diagnosis. RT Image **17(41)** (2004)

Acta Cybernetica **19** (2009) 17–40.

Improved Topic Identification for Similar Document Search on Mobile Devices*

Kristóf Csorba† and István Vajk†

Abstract

This paper presents a novel, two level classifier ensemble designed to support document topic identification in mobile device environments. The proposed system aims at supporting mobile device users who search for documents located in other mobile devices which have similar topic to the documents on the users own device. Conforming to the environment of mobile devices, the algorithms are designed for slower processor, smaller memory capacity and they maintain small data traffic between the devices in order to keep low the cost of communication. We propose a keyword list based topic comparison, enhanced with a two level classifier ensemble to accelerate the topic identification process. The new technique enables document topic comparison using few communication traffic and it requires few calculations.

Keywords: document classification, topic hierarchization, keyword selection

1 Introduction

Due to the rapid improvement of mobile devices (like mobile phones and PDAs) in storage capacity and processing capabilities, more and more information can be stored on them. People reading e-books on PDA are not unusual. There is an increasing interest in searching techniques designed for such ubiquitous environments, and most of the search engines are still based on keywords specified by the user. The techniques presented in this paper are designed for an automatic searching for documents which might be of the user's interest. The proposed system is analyzing the local documents and notifies the user if there is a document with similar topics available for retrieval. It is running as a background process and requires user interaction only if it finds something. Our work is part of a project aiming at supporting semantic search in mobile devices connected to a peer-to-peer network [1].

*This work has been fund of the Hungarian Academy of Sciences for control research and the Hungarian National Research Fund (grant number T68370).

†Budapest University of Technology and Economics, Department of Automation and Applied Informatics, Goldmann Gy. tér 3., 1111 Budapest, HUNGARY.
E-mail: {kristof,vajk}@aut.bme.hu

Searching for documents with a given topic is a frequent task today. The most common approaches are based on keywords given by the user as search criteria. Another possibility is the "topic by example" [2] approach where the user presents documents for which similar ones should be retrieved. For this purpose document topics have to be compared. This is the case in our system as well, extended with the assumption that documents stored on the users own mobile device represent the interest of the user [3].

The most common solutions for topic comparison use the bag-of-words representation of documents and calculate the similarity measure of them using the document feature vectors. These feature vectors may indicate relevance of some words to the document's topic or some extracted features. Due to the huge number of the possible words, the feature space methods usually reduce the number of features before comparing the documents. The two main approaches that use this method are feature selection and feature extraction [4][5]. Feature selection means the selection of some already available features and discarding the remaining ones, feature extraction on the other hand aims at deriving new features based on the original ones. Some feature selection methods are based on information gain or mutual information [6], least angle regression [7] or optimal orthogonal centroid feature selection [8].

If there are many document topics, using a classifier ensemble [6] may avoid the need to compare a document to every possible topic during the classification. If the number of the possible topics of a document can be limited, a classifier can be created with significantly better classification capabilities. The paper [9] proposes a technique which creates a topic hierarchy using linear discriminant projection and trains multiple classifiers, like nodes of a decision tree, to improve the classification.

The system proposed in this paper allows document classification and topic comparison of documents in mobile device environments. The main contribution consists of a feature selection algorithm and a topic hierarchy creation method. The most important property the proposed system requires is the applicability in mobile devices with limited processor and memory capacity. The created classifiers and topic comparison method have to be easy-to-calculate, and the topic comparison of documents stored on different devices has to be performed using limited communication traffic because communication between mobile devices is usually not free of charge. To conform these requirements the size and processing complexity of the document topic representations have to be in balance between very small size and good comparability. This requirement makes the frequently used huge, weighted document vectors not applicable.

The organization of the paper is as follows: section 2 presents an overview of the contribution which consists of a keyword selection and document search method presented in section 3, a two level topic identification technique presented in section 4 and experimental results presented in section 5. Finally, conclusions are summarized in section 6.

2 Overview of the contribution

Our proposed document topic identification and comparison system is based on the following key ideas:

A list of topic-specific keywords is assigned to every possible topic using a labeled training set and a supervised learning method. A document is represented by the identifier of the topic which has the most common keywords with the document, and a simple binary vector indicating the presence or absence of the given keywords in the document. This representation allows comparing the topics more detailed than just comparing the assigned topic labels of the documents.

This representation is easy to generate: the document has to be parsed and every word has to be compared to the keywords in the keyword lists. The topic with the most common keywords is selected and the binary indicator vector is created.

To create the classifier first the topic specific keyword lists have to be created: a set of the most topic-specific keywords for all possible document topics is created. Weighting of the keywords is not possible due to the binary vector in the representation which is essential for size limitations. Another important constraint implied by the proposed similarity measure (number of common keywords) is the following: keywords which would have to get negative weight cannot be used at all, because the similarity measure, the number of common keywords cannot decrease due to the presence of an additional keyword. This makes most feature selection approaches like mutual-information and information-gain based ones not applicable.

After the keyword lists have been created, the mobile devices have to identify the topic best matching their documents. They could download every keyword list and compare all of them to the documents but this would reduce scalability of the system if there were many topics. To waive this limitation, a classifier ensemble is created which is similar to a decision tree: topics are ordered into topic sets which have their keyword lists as well. This enables the mobile device to omit the check of some keyword lists and limit the search space for the best matching topic to the promising topic sets. The reason for our system not being exactly a decision tree is the following: the keyword list based topic identification may make mistakes due to the noise involved in natural language documents. A false decision inside a decision tree may make the correct classification impossible. To overcome this limitation, the topic sets are only triggering the check of their topics but not limiting the search on them. All topic sets trigger the check of their topics if their keyword lists have common keywords with the document and topic identification is performed among the triggered topics. This way, the aim of the ensemble is to exclude hopeless topics from the identification procedure and not to strictly limit the number of checked topics by always restricting the decision to the best direction on a given level of classification. This solution allows robustness against misclassifications but still reduces the number of checked keyword lists. The extension using topic sets is presented in section 4.

The search for similar documents is a very simple procedure: the mobile device downloads the compact topic representation of remote documents and calculates

the similarity, the number of common keywords, using only the representations. If it exceeds a user defined threshold, the user is notified. The user can decide if the document itself should be downloaded or not. The similarity search is presented in subsection 3.2 in details.

3 Document topic representation and comparison using keyword lists

In this section, the keyword selection method, the document representation, and the process of searching similar documents are described. As the document search is executed in the background and the user is notified when a document similar to the local ones is found, the rate of misclassifications is of key importance in this application. It is much more important than finding all similar documents.

For performance measurements we will use the common measures precision, recall and F-measure: if there is a specific target topic from which we want to select as many documents as possible, c is the number of correctly selected documents, f the number of false selections, and t is the number of documents in the target topic then precision is defined as $P = c/(c+f)$, recall is $R = c/t$, and F-measure is $F = 2PR/(P+R)$. Our aim to avoid misclassifications means the requirement of a high precision even at the price of a lower recall. But we cannot entirely omit the recall either, so we optimize F-measure while maintaining a high precision. Intuitively described a classifier with high precision is selected which is low enough to allow an acceptable recall as well.

In the description of the algorithms the following notations are used:

- $d \in T$ indicates that the document d belongs to the topic T.
- $w \in d$ indicates that the word w is present in the document d. Documents are handled as sets of words.
- $K_T = \{w_1, w_2, ..., w_n\}$ is the keyword list (set of keywords w_i) for the topic T. T might refer to a set of topics as well, in which case K_T contains all keywords of the topics in the topic set.
- A *keyword* is a word which appears in at least one keyword list. This means that keywords are words used in the topic representations.
- S_T is the selector which aims at selecting documents from the topic T. We use the selector expression instead of classifier because it selects documents for one topic, and leaves the remaining ones to be selected by other selectors. Documents not selected by any selectors will have unidentified topic. S_T is interpreted as the set of documents selected by the selector as well.
- $d \in S_T$ means that the document d is selected by the selector S_T.
- S_d is a selector based on a document d which selects documents which have common keywords with the document d.

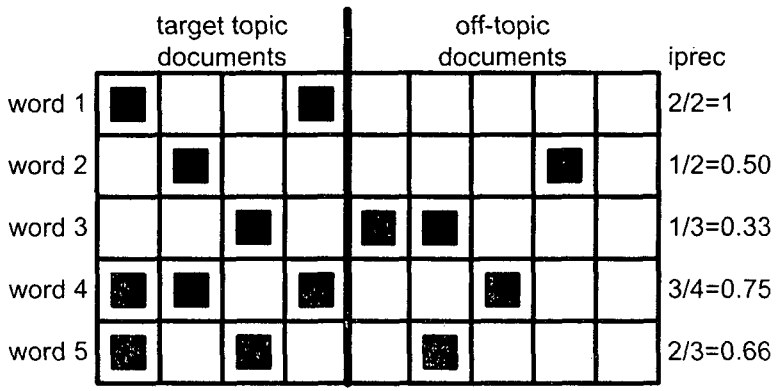


Figure 1: Example for individual precision. Rows stand for words and columns stand for documents.

3.1 Creating topic specific keyword lists

The techniques proposed in the following will require topic-specific keyword lists. These are created using the Precision-based Keyword Selection (PKS) algorithm described and evaluated in [10] in details. The PKS algorithm is based on the *individual precision* of words which is defined as follows:

Definition 1 (Individual precision). *Individual precision of a keyword w is the precision of a selector for which $d \in S_w \Leftrightarrow w \in d$, that is, it selects documents containing w.*

Fig. 1 shows an example for the individual precision. Using this quality measure of the words the PKS algorithm collects words for a K_T keyword list for a given T target topic using the following definition:

Definition 2 (Topic specific keyword). *$w \in K_T$ if and only if $P(T|w) \geq minprec$, that is, a word w is keyword of the topic T if and only if the conditional probability of the topic given w is present in a document is higher than a predefined minimal limit minprec.*

As the probability $P(T|w)$ is the expected individual precision of w, the minimal limit in the definition is called minimal precision limit *minprec*. The PKS algorithm selects topic specific keywords and selects documents containing at least one keyword. The *minprec* limit is set to allow the maximal F-measure for the selection of documents in the given topic as shown in Fig. 2.

It should be noted that there are many words in the documents which are very rare (spelling errors belong to this category too) and thus might have very high precision if their few documents belong to the same topic. To avoid such words the system does not consider words appearing in less than 0.5% of the documents in the training set. Too frequent words (stopwords) usually cannot be topic specific keywords because their expected precision is low: it is near the a-priori probability

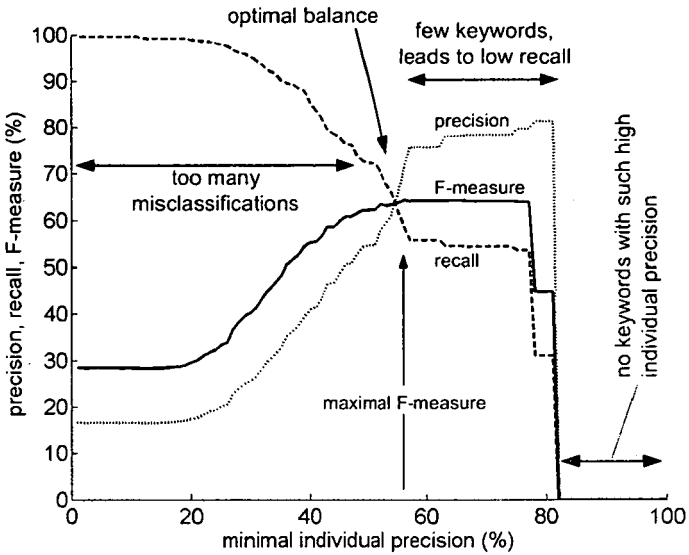


Figure 2: The PKS algorithm collects keywords starting from 100% minimal precision and decreasing it until the F-measure reaches its maximum.

of the target topic which is far lower than the *minprec* limit. In spite of this, if the number of topics is too high and it leads to a too low minimal precision, the minimal precision limit has to be limited to a predefined minimal value in order to avoid stopwords being keywords.

3.2 Searching for similar documents

Using the keyword lists created with the PKS algorithm a compact representation for every document can be created:

Definition 3 (Compact document representation). *The compact document representation of a document d is the pair $(l, \mathbf{p})$ where l is the unique identifier of the keyword list used for the representation, and $\mathbf{p}$ is the presence vector containing the binary presence indication of the keywords in the keyword list l.*

The comparison of two documents means counting the common keywords. If the two documents are represented using the same keyword list, this is a simple inner product. If the keyword lists differ, the two vectors have to be mapped into the *global keyword space* where every keyword has a unique dimension. If all the keyword lists contain the unique dimension index of the contained keywords, it can be done easily.

For the sake of simplicity, document vectors are considered in the global keyword space in the following if not stated otherwise, as these vectors are equivalent to the document representations.

If $\underline{t}$ and $\underline{d}$ are binary document (column) vectors in the global keyword space, than the similarity of the two documents is defined as

$$\text{similarity}(\underline{t}, \underline{d}) = \underline{t}^T \cdot \underline{d} \tag{1}$$

Searching for documents similar to a set of local documents is performed with the merged *base document vector*:

Definition 4 (Merged base document vector $\underline{b}$). *Merged base document vector $\underline{b}$ representing all local (base) documents for comparisons with remote documents is defined as*

$$\underline{b} := \text{sign}\Big(\sum_{d \in B} \underline{d}\Big) \tag{2}$$

where B is the set of base documents.

Given a remote compact document representation it is transformed into the global keyword space. The result is the $\underline{r}$ remote document vector. The user is notified if

$$\text{similarity}(\underline{b}, \underline{r}) \geq th \tag{3}$$

where th is the minimal similarity measure threshold which a remote document must have to the base documents in order to notify the user about its availability. The threshold is defined by the user and allows controlling the balance between precision and recall: lower threshold notifies about more documents while higher threshold notifies only if a document is really very similar to the base documents.

In order to transform a document representation to the global keyword space, the index of the keywords (their associated dimensions) are required. This information is supplied with the keyword lists themselves. If a document is represented with a keyword list not known by the mobile device, the keyword list is simply downloaded from a central repository using the keyword list identifier in the document representation. If the new keyword list has common keywords with that of the base documents, it should be stored for later use. If it is not the case, only the identifier of the keyword list should be stored in order to remember that documents using this keyword list for the document representation cannot have any common keywords with the base documents. We believe that after some initial time, unknown keyword lists will be rare. (As a possible enhancement, the central repository could tell the mobile devices which keyword lists have common keywords with a given set of keyword lists (the ones of the base documents)).

It should be noted that this type of document representation can be further improved by word stemming, part-of-speech tagging, and by adding synonyms of the keywords to the representations. Applying stemming of part-of-speech tagging would require significantly more resources. According to the extension of the documents with synonyms of the keywords, the previous paper of the authors [11] is referred to which describes a method aiming to handle synonyms and hypernyms of the keywords.

Using the methods described here, a mobile device can represent its documents for others and it can search for remote documents that are similar to the base

documents. In the following section we describe an extension which can reduce the number of keyword lists required to be checked during the topic identification process. It allows the mobile device to retrieve and store fewer keyword lists and to complete the topic identification by using fewer keyword lists.

4 Two level topic identification using topic sets

The topic identification aims at finding the topic which has the most common keywords with a given document. The keyword list of the best matching topic will be used to represent the document for other devices. A simple solution would be to calculate the number of common keywords with every available keyword list and find the best matching one. The drawback of this solution would be the high number of keyword lists: if the mobile device has to use all the keyword lists for the topic identification, it has to retrieve all possible keyword lists which decreases the scalability of the solution. In order to reduce the number of keyword lists the topic identification process has to check, a two level classifier ensemble is introduced which uses sets of similar topics on the upper level to approximate the topic of a document. Using these topic sets the number of checked topics can be limited by skipping the ones which have very low probability to be the best fitting one. In the current description we employ a two level classifier ensemble: the first level is using the topic sets and the second is using the keyword list of the triggered topics. Theoretically there is no limitation for the number of levels if the big number of topics makes more levels reasonable.

The structure of the solution is the following: during the training of the system, multiple initial topic sets are created. All of these are evaluated with a simulated document classification. This allows us to remove the useless topic sets such as those that cover almost every topic, or those achieving very low recall. In the last step of the training, final keyword lists are created for the remaining topic sets using the PKS algorithm described earlier.

During the classification, documents are first compared with the keyword lists of the topic sets. The topic sets having at least one common keyword with the document are collected (we call these topic sets the *triggered topic sets*), and finally only the keyword lists of topics in triggered topic sets (*triggered topics*) are compared to the document (Fig. 3).

The key goal of the topic set based topic identification is to limit the number of keyword lists to be checked during topic identification while not decreasing the classification performance due to the internal classifications using the topic sets.

4.1 Creating easy-to-identify topic patterns

Topic sets are created in three steps: initial topic sets are generated, initial topic sets are evaluated (and modified/removed if necessary), and further topic sets are created for every topic not covered by the topic sets. The topics not covered by topic sets are covered with separate topic sets containing only one topic.

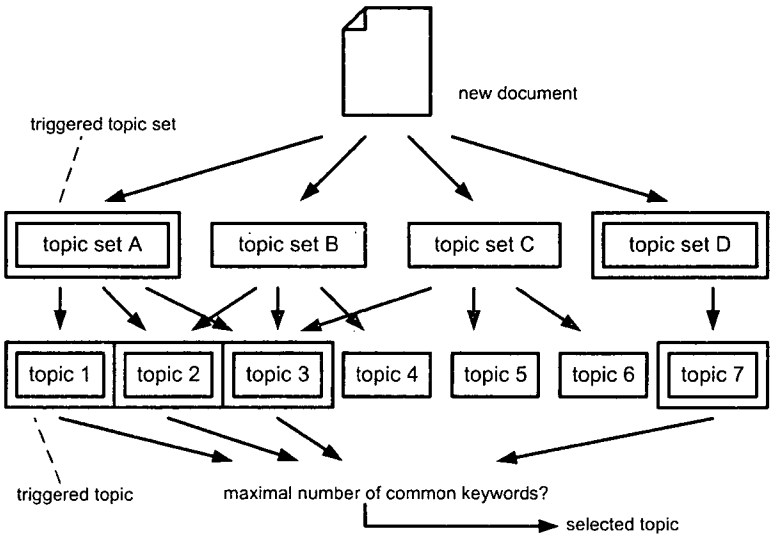


Figure 3: Topic sets. Only triggered topics (topics of triggered topic sets) are checked during topic classification.

4.1.1 Creating initial topic sets

The first step is to identify sets of topics which are easy to identify. A brute force method could be the generation of every possible subset of the topics and let the topic set evaluation step discard the bad ones. This is not applicable due to the exponential growing number of subsets. The key idea behind the *F-measure based Topic Set Creation* (FTSC) is the identification of the topic set for every w word which is the easiest to identify using only w. Similarly to the individual precision, we define individual F-measure and assign every w word that set of topics for which w achieves the highest iF individual F-measure (Fig. 4).

Definition 5 (Individual F-measure). *Individual F-measure $iF(w, \mathcal{T})$ of a w word regarding a $\mathcal{T}$ set of topics is the F-measure of a classifier selecting exactly the documents containing w.*

$$\mathcal{T}^{opt}(w) = \arg\max_{\mathcal{T}}\{iF(w, \mathcal{T})\} \tag{4}$$

Individual precision is not suitable in this case as considering more topics as target can not decrease precision. The highest precision is achieved if all the topics are target topics. A similarly defined individual recall is unsuitable as well because it does not take the precision into consideration which is still very important as we are going to create keyword lists for the topic sets using PKS.

The FTSC algorithm is searching for the topic set $\mathcal{T}^{opt}(w)$ for every w word in a greedy way: it adds the T topics to the topic set in descending $iF(w, T)$ order

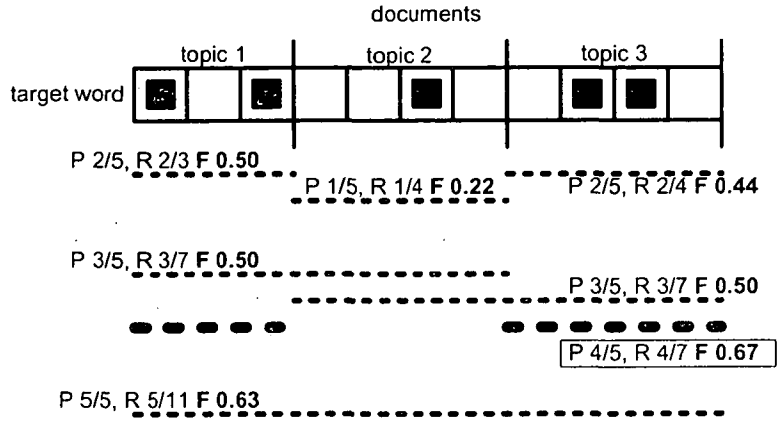


Figure 4: Example for the topic optimization for a given word. P, R and F stand for precision, recall and F-measure respectively. In this example, the individual F-measure is maximized by a topic set containing topics 1 and 3.

until the individual F-measure is maximized (Fig. 5).

The set of initial topic sets consists of all topic sets returned by FTSC executed for every w word (without duplicates of course).

Although FTSC is a greedy algorithm, it achieves optimal solution if the a-priori topic probabilities are equal for all topics:

Proposition 1. *The FTSC algorithm selects the optimal topic set $\mathcal{T}(w) = \mathcal{T}^{opt}(w)$ for every word if the a-priori topic probabilities are equal for all topics.*

4.1.2 Evaluating and modifying initial topic sets

After the initial topic sets have been created, they have to be evaluated because some of them will not be useful. For example, if a topic set covers all topics, we cannot take advantage of it. In order to use (or evaluate) a topic set its keyword list has to be created. This is done using PKS just as it would be a single topic: PKS searches for keywords which appear often in the documents of the topic set and rarely in the documents outside the topic set.

The evaluation phase evaluates every initial topic set. It creates keyword lists to distinguish them using PKS, and simulates the classification of every document in the training set. The keyword list created for an ideal topic set would select exactly the documents of the topics contained in the topic set. But topic sets may be overlapping and keywords may cause misclassifications as well. The precision and recall of the resulting classification is calculated and topic sets fulfilling the following conditions are preserved:

- Sufficiently high precision and recall. If a topic set has too low precision or recall, it is discarded. In our experiments, the minimal limit was set to 0.5

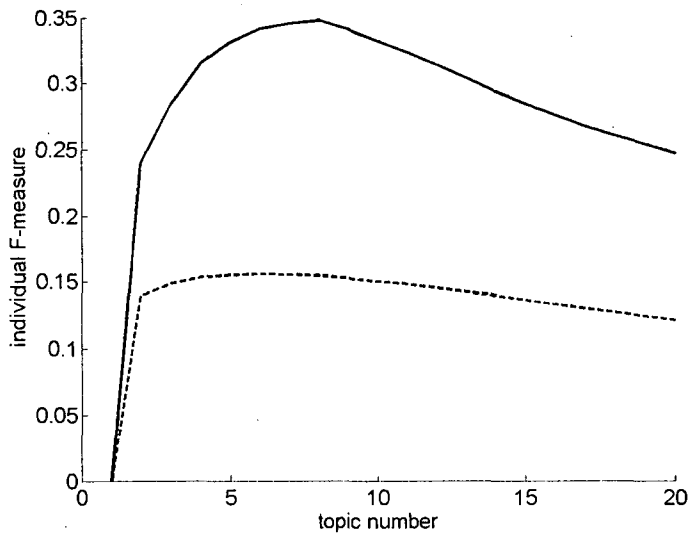


Figure 5: Evolution of individual F-measure in the 20 Newsgroups data set while increasing the number of target topics, presented for two example words. The size of the optimal topics sets are 8 and 6 in this case.

for both precision and recall.

- It cannot contain topics for which too few document were selected. Such topics are removed from the topic set because they have too low recall. The topic set would unnecessarily trigger these topics every time the topic set is triggered. In our experiments every topic had to have a 0.6 recall inside the topic set.

- The topic set has to have topics satisfying the previous condition. If all the topics of a topic set are removed due to the previous condition, the topic set is removed entirely.

- The topic set may not cover more than 50% of the topics. Otherwise there would be topic sets covering almost all topics using very common words. We believe that such topic sets are useless because they trigger almost every topic, and thus, do not support the exclusion of topics having minimal chance to be the best fitting one.

4.1.3 Creating additional topic sets

In the last step of FTSC, topics not covered by any remaining topic sets are moved into a separate topic set created for each of them individually. These additional

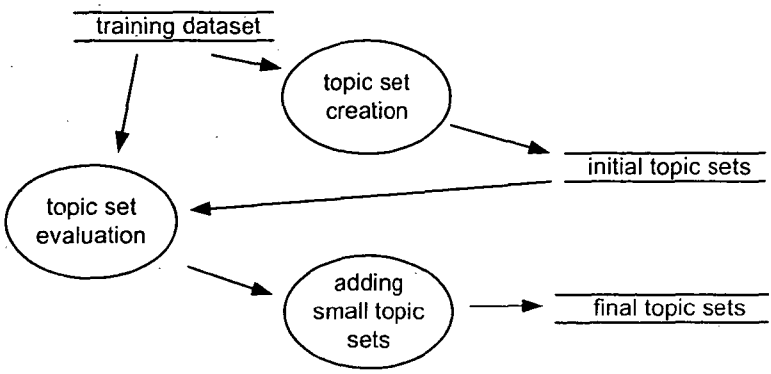


Figure 6: Data flow diagram of the FTSC algorithm. The training set is used to create the initial topic sets and the evaluation phase creates the final topic sets by modifying or removing worse topic sets and adding new ones if necessary.

topic sets contain only one topic. These topic sets are called *small topic sets* and the ones containing multiple topics *big topic sets.*

The data flow diagram of the FTSC algorithm is presented in Fig. 6.

4.1.4 Training the classifier ensemble

After the topic sets have been created, a keyword list is created for them using PKS just as they would be the topics. The second level of the ensemble is trained just as there would be no topic sets: a keyword list is created for every topic independently.

The trained first level of the classifier consists of a set of pairs

$$\{\mathcal{T} = \{T_1, T_2, ..., T_n\}; K_{\mathcal{T}}\} \tag{5}$$

where the first element is the set of topics contained in the current topic set and the second element is the keyword list for the topic set.

4.2 Using the classifier ensemble

After the training of the classifier ensemble (creating the keyword lists for all topic sets and topics) the classifier is ready to identify the topic of new documents.

If the topic of a new document has to be identified, it is first compared with the keyword lists of the topic sets. If the document has at least one common keyword with a topic set (that means that the topic set is triggered) the topics contained in the topic set are all triggered. After checking every topic set the best fitting topic specific keyword list is searched just as in the case without the topic sets, however not triggered topics are not checked because they are considered to be "hopeless".

Unfortunately there are always topics which are not covered by the initial topic sets and they have to be placed in a topic set containing only one topic. These are the *small topic sets* mentioned in the previous section. As this may increase

the number of topic sets significantly, the following rule has been introduced: if a document triggers at least an mb minimal number of *big topic sets* (containing more than one topic), the *small topic sets* are not checked because we assume that the real topic of the document is covered by the big topic sets. We call this extension *Small Sets on Demand* (SSD) because small topic sets are only checked if there seems to be a need for it.

Formally, given the d document, the $Trig(d)$ set of triggered topic sets contains the topics sets having common keywords with the document.

$$Trig(d) = \{\mathcal{T} : |K_{\mathcal{T}} \cap d| \geq 0\} \tag{6}$$

The Small Sets on Demand means

$$Trig'(d) = \{\mathcal{T} \in Trig(d) : |\mathcal{T}| \geq 1\} \tag{7}$$

and $Trig'(d)$ is used instead of $Trig(d)$ if $|Trig'(d)| \geq mb$ where mb is the minimal number of triggered big topic sets to skip the small topic sets entirely.

The $C(d)$ set of topics to check is the union of all topics in the triggered topic sets:

$$C(d) = \bigcup_{\mathcal{T} \in Trig(d)} \mathcal{T} \tag{8}$$

And finally the identified topic of the document d is the topic with the most common keywords with the document among the checked topics:

$$T(d) = \arg\max_{T \in C} \{|K_T \cup d|\} \tag{9}$$

5 Measurements

This section presents measurement results according to various aspects of the topic set based document topic identification and the search for similar documents. The measurements were performed using the commonly used data sets 20 Newsgroups [12] and the Reuters Corpus Volume 1 (RCV1, LYRL2004 split) [13].

First we present results according to the classifier ensemble used for topic identification in the 20 Newsgroups data set because the interpretation is easier with this data set. After that we present the evaluation on RCV1 and finally we present measurement results about the searching for similar documents.

5.1 Evaluation of the classifier ensemble

The most important condition the two level classifier has to satisfy is the minimal degradation in the classification performance. Table 1 presents the classification performance of the system without the application of topic sets, with topic sets but without SSD and with SSD using mb minimal triggered big topic set number 1 and 2.

From Table 1 we can draw the following conclusions:

- The case without topic sets is the baseline measurement as this is a simple classification using the keyword lists created with PKS.
- Using topic sets does not significantly influence the classification results, but without SSD, all the 13 topic sets are checked for every document, followed by the check of the triggered topics. The number of triggered topics (mean value is 3.12) is presented in Fig. 7. This means that around 16-17 keyword lists are still compared to the documents which is almost the number of topics (which is 20), thus it does not lead to significant improvement.
- By activating SSD the classification performance decreases slightly because some documents belong to topics in small topic sets but still trigger enough big topic sets which makes their real topics not checked. But for an exchange, with $mb = 2$, altogether 54% of the documents with topics in big topic sets is classified without checking the small topic sets (thus checking only $5 + 3.12$ keyword lists in average).
- SSD with $mb = 1$ decreased the recall slightly more but it made the small topic sets skipped for every document which had a real topic in one of the big topic sets.

Table 1: Classification results, with the 20 Newsgroups data set, with and without topic sets (no Small Sets on Demand, every topic set is always checked) and with SSD using mb (minimal number of triggered big topic sets to skip checking of the small topic sets) 1 and 2.

	precision	recall	F-measure
without topic sets	0.61	0.45	0.50
with topic sets	0.65	0.42	0.50
SSD ($mb = 2$)	0.64	0.41	0.49
SSD ($mb = 1$)	0.65	0.39	0.47

If a user stores documents belonging to big topic sets on the mobile device, setting $mb = 1$ can decrease the number of keyword lists compared to the document during topic identification from 20 (no topic sets, no SSD) to 8.12 in average. If the small topic sets are needed as well, this value is 16.12 in average.

Details about the topic sets are presented in Table 2. Some topic sets seem to be reasonable based on the name of the contained topics like merging *atheism* and *religion.christian*. Others may look strange in the first approach but after having a look at some keywords assigned to these topic sets, a connection can be recognized. Topic set 1 is based on connections with security and nation names, topic set 2 is about sports but nation names lead to the topic on the middle east as well. Topic set 3 is clearly about X-servers and MS-Windows, topic set 4 is based on security aspects of politics and computer science, and finally topic set 5 is clearly about

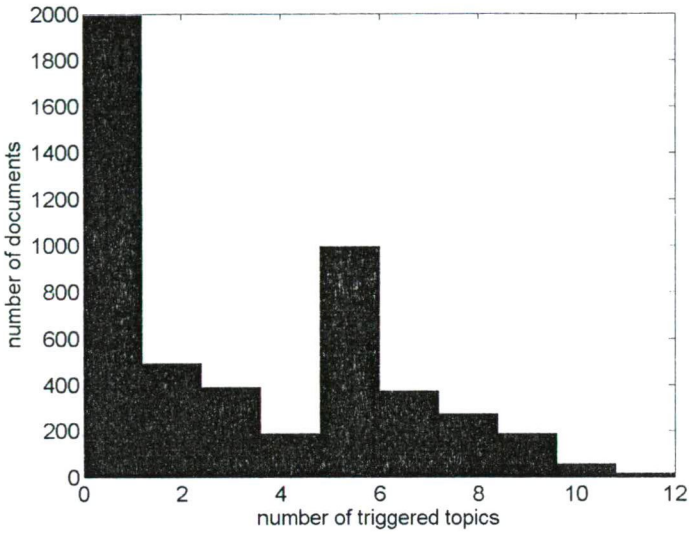


Figure 7: Histogram of the number of triggered topics in 20 Newsgroups. The mean value is 3.25 topics.

religions. Small topic sets are not mentioned here but every topic not covered by the presented topic sets is covered by a small topic set.

If we evaluate the FTSC method as a method for ordering topics into hierarchy, we can see that the created topic hierarchy is not the same as the original topic hierarchy of the 20 Newsgroups data set. The main reason for this difference is that the resulting "hierarchy" is created by merging topics which can be easier recognized using keywords if they are merged, than if they would have to be recognized separately. This is caused by many common potential keywords shared between the documents of the topics. As there are many keywords it is not surprising that some of them suggest different merging of topics than the merging defined by the original topic hierarchy of the data set. For example topic set 5 contains "alt.atheism" and "soc.religion.christian" together and it is reasonable as well although the original hierarchy does not indicate this similarity.

The covering of topics by topic sets is visualized in Fig. 8. During the application of the system, documents of a given topic may trigger multiple topic sets. The corresponding measurement results are presented in Fig. 9. The covering of the topic sets can be clearly recognized but there are false triggers as well. The average number of topic sets a document is triggering is 1.23, its histogram is shown in Fig. 10.

Table 2: Topic sets in 20 Newsgroups. Quality is shown in terms of precision (P) and recall (R). Topics not covered by any sets mentioned in this table have their own small topic set.

ID	contained topics	quality	example keywords
1	soc.religion.christian	P 57	christians church pgp soviet
	talk.politics.mideast	R 70	muslim heaven spirit
	sci.crypt		secret israeli secure arab
	sci.space		security orbit encryption
2	rec.sport.baseball	P 71	teams team turkish baseball
	rec.sport.hockey	R 65	season hockey player fans nhl
	talk.politics.mideast		league players israeli arab
3	comp.os.ms-windows.misc	P 62	server microsoft window
	comp.windows.x	R 68	motif
4	talk.politics.guns	P 63	cars pgp citizens cup economic
	rec.sport.hockey	R 69	guns secure wings federal fbi
	sci.crypt		warrant security weapons keys
	rec.autos		enforcement hockey coverage
	talk.politics.misc		gun criminal crime secret nhl
			car police encryption agents
5	alt.atheism	P 73	atheist christians bible holy
	soc.religion.christian	R 70	belief morality sin heaven god
			church faith

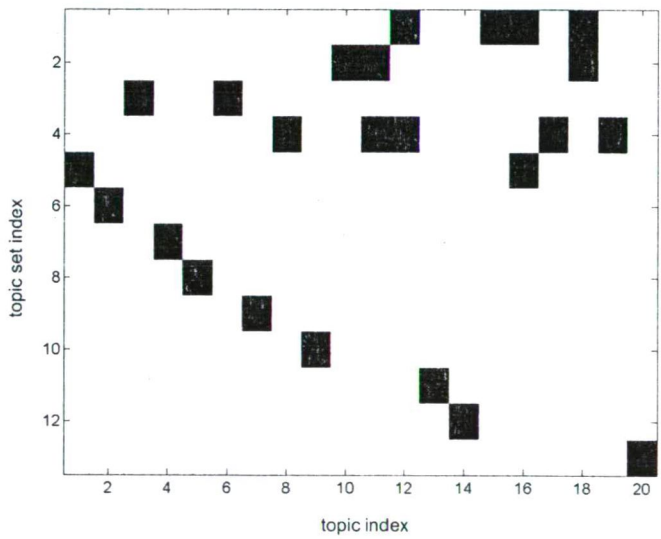


Figure 8: Topic sets retrieved for the 20 topics of the 20 Newsgroups data set.

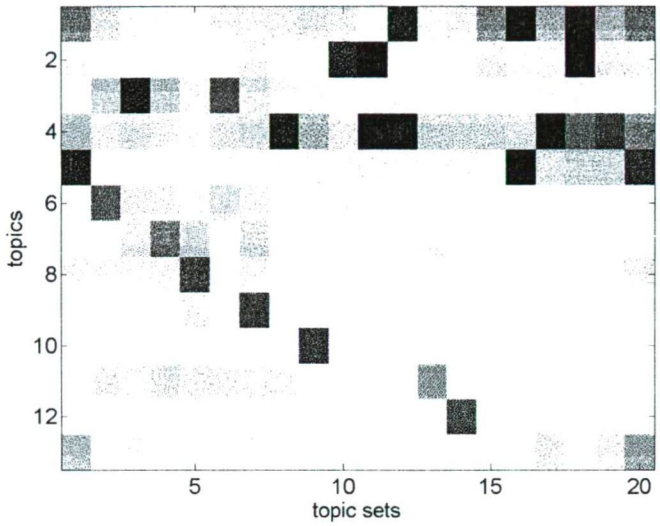


Figure 9: Triggering of topic sets in 20 Newsgroups. The more often a topic set is triggered by the documents of a topic the darker is the rectangle corresponding to the (topic set;topic) pair. The measurement used SSD with $md = 2$.

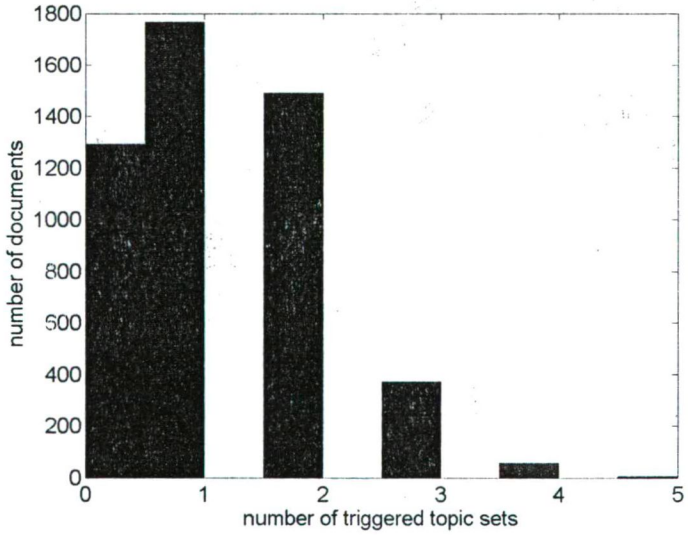


Figure 10: Histogram of the number of triggered topic sets in 20 Newsgroups. Mean value is 1.23 topic sets.

5.2 Evaluation on Reuters Corpus Volume 1

The 20 Newsgroups data set has only 20 topics. The Reuters Corpus Volume 1 (version 2) has 103 topics altogether and these are organized in a two-level hierarchy containing 4 topics on the upper level. The LYRL2004 split of the data set which we used for the measurements has an already prepared word-document matrix available on the World Wide Web. This prepared version of the data set has stemming already applied to it.

During the preparation of the data set we removed the predefined upper level topics *CCAT*, *GCAT*, *MCAT* and *ECAT* from the data set, and some further topics which contained too few documents were removed too. 78 topics remained.

We applied exactly the same methods to the RCV1 data set as to the 20 Newsgroups previously. The classification results were obtained without topic sets and with topic sets using SSD with $mb = 2$. The results presented in Table 3 are similar to the ones for the 20 Newsgroups (Table. 1). We believe that the small decrease in performance with topic sets is caused by the imbalanced training set as the number of documents in the various topics in RCV1 is not constant.

Table 3: Classification results on the RCV1 data set.

	precision	recall	F-measure
without topic sets	0.64	0.41	0.47
with topic sets			
SSD ($mb = 2$)	0.62	0.47	0.52

As RCV1 has much more topics than 20 Newsgroups, the capability of the topic sets to decrease the number of keyword lists checked with a given document during topic identification is more significant: although there are 78 topics, the mean number of triggered topics is 37.8. If no small topics are required to check, only 12 big topic sets are checked and 4.7 of these big topic sets are triggered by a document in average. This means that if there is no need to check small topic sets, the classification of a document requires the check of 12 big topic sets and those trigger 37.8 topics in average, so $12 + 37.8 = 49.8$ keyword lists are checked in average, instead of 78.

Examples on the merged topics and keyword lists of the topic sets are presented in Tables 4 and 5. Due to the high number of topics we cannot present every topic set with all its keywords. Incomplete lists are marked with "...". There are many words which are rare enough not to be discarded as stopwords but they imply topic sets containing lots of topics. This leads to some topic sets (ID 10, 11 and 12) which have too many topics and thus too many and very diverse keywords as well. Although they were triggered by over 80% of the documents, they do not contain more than 50% of the topics so they were not discarded. Due to space limitations these 3 topic sets are not described in the table.

Based on tables 4 and 5 the topic sets have clearly captures some similarities between the merged topics: sometimes it conforms the original hierarchy like topic sets 3 and 4, and sometimes it captures other similarities like topic set 5 containing

marketing, strategy and performance measurement together, or topic set 2 merging sports with related markets.

Table 4: Topic sets in RCV1. The topics are represented by their code name in the RCV1 data set. The first letter identifies the four upper level topics *corporate/industrial*, *economics*, *government/social* and *markets*.

ID	topics
1	M11, C15
2	M14, GSPO
3	GPOL, GDIP
4	M14, M11
5	M14, C15, M11, M13, M12, E12, C18, C11, C31
6	GCRIM, C15, GPOL, GPRO
7	GCRIM, GPOL, C12
8	M11
9	GPOL, M14, GSPO
10	C15, GSPO, M14, GPOL, GDIS, GCRIM, M11, C21, GDIP, M13, E12, C11, GWEA, GVIO, E21, E11, C42
11	M14, C15, M11, M13, GPOL, GCRIM, C18, C13, GDIP, C17, E21, C11, M12, GSPO, GVIO, C21, E12, C24, C12, E51, C42, C31, C41, GPRO, C33, GDEF, GDIS, E11, C22, G15, E13, E41, C14, GENV, C16, GHEA
12	C15, M14, M13, GPOL, C31, GCRIM, M11, C21, GDIP, E12, C13, GVIO, C11, M12, C18, E51, E11, E71

5.3 Evaluation of similar document search

In order to have an overview on the task of searching for similar documents first we identified the topic of documents in the 20 Newsgroups data set using the techniques discussed before. Using the document representations we calculated the document similarity matrix to have an overview on the similarity structure. The matrix is presented in Fig. 11. If we used a single base document which corresponds to a row (or column) vector in the matrix and the similarity threshold would be $th = 1$, the set of selected documents would be the set of documents indicated by dots in the figure.

In order to evaluate the search method we simulated it using various settings:

- The number of base documents were 1, 5, 10, 15 and 20.
- Threshold values between 1 and 10 were investigated.
- The base documents were always taken from one topic, but every 20 topics were investigated this way.

- For every setting of the previous parameters 100 measurements were performed by selecting random sets of base documents with the given size.

Table 5: Topic sets in RCV1: the automatic reconstruction of the topic hierarchy. Percentage under the topic identifier shows the ratio of documents triggering this topic set. Keywords without proper ending are a result of the stemming applied to the data set.

ID	topics	example keywords
1 23.6%	equity markets, performance	pretax, dax, pfennig, outperform, payout, canon, goldfield...
2 13.6%	commodity markets, sports	cup, cricket, medal, coach, sheffield, yorkshir, wimbledon, athlet, mideast, lbs, intermonth, goalkeep, unbeat, semifin...
3 8.0%	domestic politics, international relations	podium, obstruc, diplom, palestin, gaza, elector, bosnian, sworn, boycot, disarm, peacemak, provoc, breakaway, proclaim...
4 13.6%	commodity markets, equity markets	unlead, gallon, composit, backward, meal, mideast, lbs, intermonth, overbought, telefon, sunseed, backfat, cottonseed...
5 61.9%	commodity markets, performance, equity markets, money markets, bond markets, monetary/economic, ownership changes, strategy/plans, markets/marketing	volum, benchmark, stead, technic, buy, profit, actual, commod, mercantil, pork, factor, unlead, gallon, chip, unchang, liquid, yen, outweigh, pfennig, underperform, platin, bombay, payout, midpoint, interbank, forint, overvalu, oversold, cottonseed, financier, hectic, nationsbank...
6 29.4%	crime, law enforcement, performance, domestic politics, biographies, personalities, people...	widow, kidnap, jail, convict, extraordin, cocain, crim, murd, amnest, interpol, imprison, mafia, heroin, theft, horror, cardiac, bodyguard...
7 13.9%	crime, law enforcement, domestic politics, legal/judicial	courtroom, conspir, kidnap, jail, corrupt, truth, crimin, fbi, arrest, interpol, heroin, motorway, bodyguard...
8 6.0%	equity markets	chip, composit, fts, nikkei, dax, ibi, cac, seng, komercn, sse, nse, tisc, telefon...
9 14.6%	domestic politics, commodity markets, sports	cargoe, cricket, halftim, medal, coach, wimbledon, athlet, octan, goalkeep...

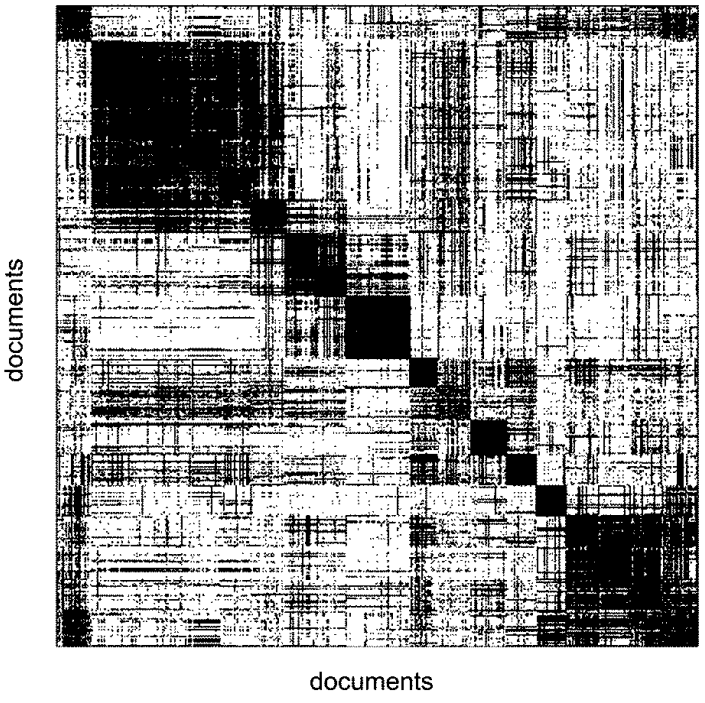


Figure 11: Document similarity matrix of 20 Newsgroups. Dots indicate non-zero similarity of the documents corresponding to the row and column. As the documents are ordered by topics along the axes topics are visible as rectangles along the main diagonal because documents inside the same topic tend to contain common keywords.

The results are presented in Fig. 12. It is clear that increasing the threshold increases precision and decreases recall as less documents have the chance to be selected and documents from the target topic (topic of the base documents) have usually more common keywords with the investigated remote document. Using one single base document provides very few keywords for the similarity search thus it leads to low recall values although with higher precision. More base documents provide more keywords, thus, it increases the recall but leads to more misclassifications as well, because more keywords introduce more chances for false classifications. As the threshold is defined by the user the balance between high precision and low recall (few false notifications but few retrieved documents), or lower precision and high recall (more mistakes but more found documents) can be set according to the preferences of the user.

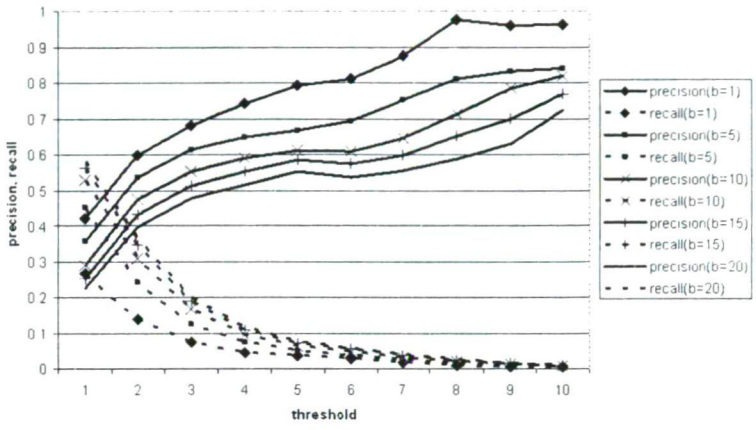


Figure 12: Results of the search for similar documents. Precision and recall is presented for various b number of baseline documents in the function of the th threshold.

6 Conclusions

We proposed a similar document retriever system which employs topic specific keywords to create compact topic representations allowing topic comparisons without downloading a whole document. The topic identification is an important step during the creation of the compact document representation. The improvement of this step presented in this paper allows the topic identification to download and process less possible topics. This accelerates the procedure and reduces communication traffic. The details of the applied algorithms and experimental results were presented in this paper.

Acknowledgements: This work has been fund of the Hungarian Academy of Sciences for control research and the Hungarian National Research Fund (grant number T68370).

References

[1] B. Forstner and H. Charaf. Neighbor selection in peer-to-peer networks using semantic relations. *WSEAS Transactions on Information Science and Applications*, Volume 2(Issue 2):239–244, February 2005. ISSN 1790-0832.

[2] W. Buntine. Topic-specific scoring of documents for relevant retrieval In *Proceedings of ICML 2005 Workshop 4: Learning in Web Search*, 7 August 2005, Bonn, Germany, 2005.

[3] Paul-Alexandru Chirita and Claudiu S. Firan and Wolfgang Nejdl. Summarizing local context to personalize global web search. In *CIKM '06: Proceedings of the 15th ACM international conference on Information and knowledge management*, pp. 287–296, Arlington, Virginia, USA, 2006., ACM Press.

[4] Sholom M. Weiss, Nitin Indurkhya, Tong Zhang, Fred J. Damerau, editor. *Text Mining, Predictive Methods for Analysing Unstuctured Information.* Springer, 2005.

[5] B. Fortuna, D. Mladenić, and M. Grobelnik. Semi-automatic construction of topic ontology. In *Proceedings of SIKDD 2005 at multiconference IS 2005*, Ljubljana, Slovenia, 2005.

[6] L. R. Oded Maimon, editor. *The Data Mining and Knowledge Discovery Handbook.* Springer, 2005.

[7] S. S. Keerthi. Generalized lars as an effective feature selection tool for text classification with svms. In *Proceedings of the 22nd International Conference on Machine Learning, Bonn, Germany*, 2005.

[8] J. Yan, N. Liu, B. Zhang, S. Yan, Z. Chen, Q. Cheng, W. Fan, and W.-Y. Ma. Ocfs: optimal orthogonal centroid feature selection for text categorization. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference*, pp. 122–129, New York, NY, USA, 2005. ACM Press.

[9] Tao Li, Shenghuo Zhu, and Mitsunori Ogihara Hierarchical document classification using automatically generated hierarchy In *Journal of Intelligent Information Systems* Springer Netherlands, Volume 29, Number 2, 2007.

[10] K. Csorba and I. Vajk. Supervised term cluster creation for document clustering. *Scientific Bulletin of Politehnica University of Timisoara, Romania, Transactions on Automatic Control and Computer Science*, Vol. 51, 2006.

[11] K. Csorba, I. Vajk Improving Document Similarity Measurement for Mobile Environment with Document Extension In *ECML PKDD 2008, Ubiquitous Knowledge Discovery Workshop* Antwerp, Belgium, 2008. `http://www.ecmlpkdd2008.org/`

[12] K. Lang. NewsWeeder: learning to filter netnews. In A. Prieditis and S. J. Russell, editors, *Proceedings of ICML-95, 12th International Conference on Machine Learning*, pages 331–339, Lake Tahoe, US, 1995. Morgan Kaufmann Publishers, San Francisco, US.

[13] Lewis, D. D.; Yang, Y.; Rose, T.; and Li, F. RCV1: A New Benchmark Collection for Text Categorization Research. In *Journal of Machine Learning Research*, 5:361-397, 2004. `http://www.jmlr.org/papers/volume5/lewis04a/lewis04a.pdf`.

Appendix: formal proof of proposition on optimality of topic set selection in FTSC

Proposition. *The FTSC algorithm selects the optimal topic set* $\mathcal{T}(w) = \mathcal{T}^{opt}(w)$ *for every word if the a-priori topic probabilities are equal for all topics.*

Proof. We consider a given word which selects documents of a given target topic. Let c be the number of correctly selected documents, s the number of selected documents, and t the number of documents in the target topic. The precision is $p = c/s$ and recall is $r = c/t$.

$$F = \frac{2 \cdot p \cdot r}{p + r} = \frac{2 \cdot c \cdot c}{s \cdot t(c/s + c/t)} = \frac{2 \cdot c}{t + s} \tag{10}$$

If we search for the optimal topic set for a given w word, the s number of selected documents is constant. We assume that the t target document number is the same for every topic (assuming equal a-priori topic probability). A topic set containing $|\mathcal{T}| = n$ topics and maximizing F-measure is maximizing

$$F = \frac{2 \cdot \sum_{T \in \mathcal{T}} c_T}{n \cdot t + s} \tag{11}$$

where c_T is the number of selected documents in the topic T. Due to the constant denominator, $\mathcal{T}$ has to maximize $\sum_{T \in \mathcal{T}} c_T$. Considering that the individual F-measure of w in every topic is

$$iF(w, T) = \frac{2 \cdot c_T}{t + s} \tag{12}$$

where the denominator is topic independent, $\mathcal{T}$ has to contain the n topics with the highest individual F-measure regarding w. If we add the topics to $\mathcal{T}$ in decreasing individual F-measure order, the $\mathcal{T}$ maximizing F-measure is a global optimum. □

Acta Cybernetica **19** (2009) 41–60.

3-level Confidence Voting Strategy for Dynamic Fusion-Selection of Classifier Ensembles

Csaba Főző and Csaba Gáspár-Papanek*

Abstract

There are two different stages to consider when constructing *multiple classifier systems*: The *Meta-Classifier Stage* that is responsible for the combination logic and basically treats the ensemble members as black boxes, and the *Classifier Stage* where the functionality of members is in focus. Furthermore, on the upper stage - also called voting strategy stage - the method of combining members can be done by fusion and selection of classifiers. In this paper, we propose a novel procedure for building the meta-classifier stage of MCSs, using an *oracle* of three-level voting strategy. This is a dynamic, half fusion-half selection type method for ensemble member combination, which is midway between the extremes of fusion and selection. The MCS members are weighted and combined with the help of the oracle, which is founded on a voting strategy of three levels: (1) The Local Implicit Confidence (LIC), (2) The Global Explicit Confidence (GEC), and (3) The Local Explicit Confidence (LEC). The first confidence segment is dependent of classifier construction, via the implicit knowledge gathered simultaneously with training. Since this strongly depends on the internal operation of the classifier, it can not always be obtained, for example, when using some particularly complex classification methods. We used several, known classifier algorithms (Decision Trees, Neural Networks, Logistic Regression, SVM) where it is possible to extract this information. The second goodness index is calculated on the validation partition of the labeled train data. It is used to obtain the general accuracy of a single classifier using a data set independent of the training partition. And finally, the third part of the confidence triplet depends also on the unlabeled objects yet to be classified. Due to this, it can only be calculated in classification time.

Keywords: multiple classifier systems, supervised learning, classifier ensembles, voting strategies, confidence based voting, two-staged ensemble

*Budapest University of Technology and Economics, Department of Telecommunication and Media Informatics, Telephone: +36-1-463-2225, Fax: +36-1-463-3107, E-mail: {csaba.fozo, gaspar}@tmit.bme.hu

1 Introduction

One of the most promising ways of fighting the challenge of *supervised learning* in artificial intelligence, or classification as it is mostly known in the lingo of data mining, is using *classifier ensembles*. Classification is the most significant type of analysis in data mining [18], and thus an active area of research as well. More than 10 years ago a new performance increasing technique emerged by combining classifiers. Many studies have been published about the advantages of the paradigm over the individual classifier models [5, 14]. This seems to be a quite easy and widespread way of further increasing classification accuracy, although, beyond the basic idea there are also several new challenges of key importance, like preserving ensemble members' accuracy while increasing their diversity [11].

The aim of this paper is to present a new combination method for *Multiple Classifier Systems* (MCS) [16, 17, 13] to further increase accuracy of classifier ensembles. Thus, this novel method is applicable to any MCS, that uses a separated algorithm to handle the MCS members' individual results as inputs, and provides a single, final result as an output. Our method treats the underlying single classifiers as black boxes. The only requirement for the individual classifiers is to be able to provide an implicit confidence value measuring the sureness of their decision. By dividing ensembles to a classifier and a meta-classifier stage, it's possible to build a pure meta-classifier stage algorithm, such as the 3 level confidence voting method, that has the advantages of transportability.

Our method has three levels of confidence, GEC, LIC and LEC. In understanding the importance of each level, a parallel can be drawn between this voting technique and a group of specialists at a meeting. The process of classification would be paired with a discussion in the example. In case of a single classifier, the group would consist of only one specialist, who makes a decision. With more than one members, a chairman has the final word. The chairman would be the voting algorithm. The more information it has about the members, the easier to make the best final decision. In the three-level voting method, this information is materialized in the three levels. The global explicit level refers to a statistical information about the given specialist's general accuracy when making decisions, based on past situations observed by the chairman. The local implicit level is a self-confidence-like value, describing how sure he really is about his answer, which can have a significant influence on how persuasive he is at convincing the chairman. It's not enough to know the correct decision, but we have to know, how sure he is about it, e.g. when two, usually very accurate specialists decide differently, the chairman has to size up the situation taking into account which one of them is more confident. And finally the local explicit level weighs the specialists by separating the question space into fields and determine how accurate the specialist generally is on the field of the given matter statistically. This way it helps identifying the level of diversity between MCS members, and assigning greater weights to the ones that are having more expertise on the field.

The procedure is similar to [8], with the difference that we are interested in precise confidence values to select and weight members' decisions in the combination,

instead of the best single classifier for an unlabeled data point. As we could see in the example of the previous paragraph, neither of the three confidence levels can be omitted. Even with perfectly specialized members and complementary expertise, we have to take into consideration that the field of the question, the given subject matter can not be identified perfectly. The classifiers wouldn't make a good decision, if the subject matter is, for example, near the boundary of two specialists expertise. In this case, using only the local confidence can give a quite inaccurate weighing. Solving the problem is even harder in real situations, when specialist's expertise is always overlapping, and nor the matters can be clearly assigned to a specialist, neither the real expertise on the field can be derived. The error of locating the given subject can distort the measured LEC value substantially. In conclusion, the imperfection of finding the question's local region results that, at these hard cases, a good concept to follow is to complete local confidence with a general decision accuracy, such as the global explicit confidence. In fact, if we only wanted to select the best specialist to make the decision, like Giacinto et al. [8], this wouldn't be such an important matter, however, our approach can provide significant improvement in classification accuracy.

The basic idea of combining these three levels of confidence is similar to that used at building classifier ensembles instead of individual classifiers. We combine several confidence values of each single classifier, to determine the best weighing amongst them, resulting the best combination. It is like using an other ensemble system to support the whole classifier ensemble. The three-level voting technique applied to an ensemble of classifiers can be described as an MCS with two ensembles, one traditional ensemble of the classifier stage, and a second one of the meta-classifier (voting) stage. The first one summarizes the members' decisions, and the second one summarizes the members' confidence indices to provide weights for the first one.

The following section offers a brief oversight of classifier ensembles and a possible categorization. Section 3 presents the three-level confidence voting technique. The dataset used for experimental testing and the results are reviewed in Section 4. And finally Section 5 concludes the paper by summarizing retrieved experiences, furthermore outlining possible directions of future work.

2 Classifier Ensembles

One of the main reason of building classifier ensembles was that many classifiers that were considered weak was found to keep a significant part of their capabilities hidden if applied individually. Even the most simple algorithms, like decision trees' performance can be boosted by using a properly collected ensemble [19, 13]. The fundamental thought is that the precision of an individual classifier can be superseded by the diversity of a classifier ensemble. The key idea is similar to the theory of division of labour. In some ways, diversity is identifiable with specialization [15, 11].

Definition 1 (Classifier Ensemble). *Let* $\overline{x} = [x_1, ..., x_N]$ *be a data point described by* N *features and let* $\boldsymbol{X} = [\overline{x}_1, ..., \overline{x}_n]^T$ *be the training dataset in a form of an* $n \times N$ *matrix. Let* $\overline{y}_{\boldsymbol{X}} = [y_1, ..., y_n]^T$ *be a vector of class labels for the training data, where* y_i *takes a value from* $C = \{l_1, ..., l_c\}$ *class label set. Let* L *be the number of classifiers in the ensemble denoted by* $D_1, ..., D_L$. $D_i(\boldsymbol{X}, \overline{y}_{\boldsymbol{X}}, \overline{x}) = d_i$ *is the decision of* D_i *classifier ensemble member, taking a value from* C. *Then* E *is a Classifier ensemble, if*

$$E = \{D_1, ..., D_L, V\}, \tag{1}$$

where

$$V = V(d_1, ..., d_L) \tag{2}$$

is the voting logic upon the D_i *ensemble members.*

2.1 Categorization

Many dimensions can be used to categorize classifier ensemble systems. Different approaches are known on how classifiers should be collected (1.), what are the construction terms (2.), in what manner should they work together (3.), and how the final output of the ensemble should be created based upon the members' decisions, i.e. the voting strategy (4.).

The member collection can be separated mainly into two large groups. In the first segment, the same kind of classifiers are used in the ensemble, and the diversity factor depends totally on the different methods of training the model. $E = \{D_1, ..., D_L, V\}$, where $D_i = D(X_i, Y_i, x)$. On the other hand, the dissimilarity of the ensemble can be largely enhanced by selecting members, based on different core algorithms. In certain terminologies, only the firstly mentioned collection is called a classifier ensemble, while the lingo of data mining is still unifying in this area, thus, from a more general view-point, any multiple classifier system (containing more than one classifier) can be called this way.

The terms of construction also consist of two possible directions. Considering dependencies between ensemble members, the single classifiers can be independent, hence the system can be built concurrently. While if there are dependencies, the construction has to be sequential.

Working manner is typically partitioned into two complementary approaches [10]: *Selection* and *fusion*. Selection follows the assumption that the members are having complementary competence on the feature space, thus in the divided space, they can be more specialized and accurate. While the underlying idea of classifier fusion is that the members are experts of the whole feature space. Due to diversity, the ensemble members may misclassify different input objects, thus a consensus type voting will filter errors. A mixture of the two methods is also possible.

Voting strategies can be various and can be heavily dependent from the above mentioned techniques also. Consensus type voting is generally used with fusion working manner. The simplest way is balanced majority voting. The weights used can be derived by model validation, when each member's accuracy is measured by detaching a validation segment from the training data and using it to score

Table 1: References of ensemble systems

	1		2		3		4			
	1.a	1.b	2.a	2.b	3.a	3.b	4.a	4.b	4.c	4.d
AdaBoost [7]	X			X	X			X		
MultiBoost [20]	X		X		X		X			
Bagging [4]	X		X		X		X			
RndForest [6]	X		X		X		X			
RndSubspace [9]	X		X		X		X			
RotForest [19, 13]	X		X		X		X			
RndLinOracle [12]		X	X		X	X				X
3-LevelVoting	-	-	-	-	X	X		X	X	X

aggregated performance of the particular classifier (e.g. Global explicit confidence). On the other hand, it is possible to adjust the before mentioned weights based upon the member itself, by attaching a confidence value to each of its decisions (e.g. Local implicit confidence). And finally, some systems use a higher authority called oracle to calculate members' expertise based upon the object to be classified, making uneven merging of member outputs, or even totally excluding some members (e.g. Local explicit confidence).

The uprising dimensions for characterizing ensemble systems are summarized in a list below and well-known algorithms with references are assigned to the classes in Table 1. Later, we will refer back to this enumeration to differentiate the analyzed methods to ease recognition of trends and possibly enticing fields of future work.

1. Member selection
   a) Same kind of classifiers (one learning method)
   b) Different classifiers (several learning methods)
2. Construction
   a) Concurrent
   b) Sequential
3. Working manner
   a) Selection based
   b) Fusion based
4. Voting strategy
   a) Balanced majority voting

b) Validation based average confidence
c) Self based confidence
d) Oracle type

In the following section, we apply the upper mentioned categorization on the *three-level confidence voting* as it's summarized in the last row of Table 1, and we give a detailed description of the algorithm and the three levels' logic.

3 The Three-Level Confidence Voting

As it was stated above, the three level voting technique is a meta-classifier stage algorithm. Thus, it is separable from the classifier stage, resulting categories 1 and 2 independent of our algorithm. On the 3. dimension of the system, our method falls within both 3.a and 3.b categories, since either of them can be applied. And finally, our voting strategy is a novel mixture of 4.b, 4.c, and 4.d types.

The general voting logic upon the D_i ensemble members in (2) is partly modified here, due to more than one confidence metric is used. Hence,

$$V = V_{3-level}(V_{LIC}(D_1, ...D_L), V_{GEC}(D_1, ...D_L), V_{LEC}(D_1, ...D_L)), \qquad (3)$$

where *LIC* stands for Local Implicit Confidence, *GEC* for Global Explicit Confidence, and *LEC* for Local Explicit Confidence. First we demonstrate the details of the three levels, and then the $V_{3-level}$, that combines them.

3.1 Local Implicit Confidence (LIC)

The 'local' keyword means that this is a data point specific measure, that describes the data yet to be classified, instead of describing the classifier as a whole. Thus, it can be different for each test data point. Furthermore, it is measured implicitly, inside the classifier, bearing the information of both the model, and the test data. Most of the classifier algorithms are able to provide an index like this by the trained model and the given test data. With others that are not, some modifications have to be applied first. In this paper, we only consider classification algorithms that are able to produce this measure (Decision Trees, Neural Networks, Logistic Regression, SVM). While training the model, numerous attributes, weights are set by the classifier algorithm inside the model. These influence the model's accuracy as well, but their primary objective is to predict classification's expectable correctness, solely based on training experiences. Thus, when a classifier gives a decision, e.g. the label of data point x is $D_i(x) = l_j$, then it assigns a probability to this decision as well, e.g. $LIC_{D_i,x} = conf_{D_i(x)=l_j} = 70\%$ that discribes the classifier's certainty that it is the good decision.

3.2 Global Explicit Confidence (GEC)

The GEC is greatly different from the previous index. The global explicit level is a statistical information about the given, individual classifier's general accuracy.

Global refers to the generality, which means that it is a feature of the classifier, and it is identical for all incoming data to be classified. Explicit refers to the measuring method's independence from the classifier's inner processes. Practically, it's the probability of making a correct decision, based on the average accuracy reached on the validation segment of the labeled data. After training has finished, the trained model is tested on validation data to determine the model's accuracy on an input different from the training data. This way, the model's real accuracy can be estimated much more precisely than by performance measured on the same data that it was trained on. The specific algorithm for calculating the GEC always depends on the dataset and the task to resolve. Otherwise, we would not be able to compare the results of the MCS and the individual classifiers.

3.3 Local Explicit Confidence (LEC)

The local explicit confidence, considering it's features, is somewhere between the above presented two. Practically, it's the accuracy measure describing the classifier, and the test data points as well. The latter part is somewhat similar to LIC, although it's statistically calculated, independently from the classifiers inner processes, which means that it's similar to GEC. The basic idea is to locate the training data points that are similar to the data point to be classified, and calculate the classifiers' global explicit confidence upon this set of data points. Hence, it provides an accuracy that describes the local region of the test data point. So it depends on the test data point, because it's needed to gather the local region.

First, the algorithm locates the $\overline{x}_1, ..., \overline{x}_k$ nearest neighbors (k-NN), also called the local region of $\overline{z}$ unlabeled object in the feature space, with the help of an appropriate distance metric over data objects. Naturally, this metric is always strongly dataset dependent. In case of a normalized dataset, where data points' features are numeric values without further information, using an Euclidean distance metric proves to be a good choice. In other cases, the selection has to be considered more thoroughly.

In the next step, we count the $\overline{x}^{MCB} = [d_1(\overline{x}), ..., d_L(\overline{x})]$ Multiple Classifier Behaviour [21], where $d_j(\overline{x}) = D_j(\mathbf{X}, \overline{y}_{\mathbf{X}}, \overline{x})$, containing the ensemble members' decisions, for each $\overline{x}_i$ nearest neighbor, and for the $\overline{z}$ object to be classified. The local region is further narrowed by selecting the ones, having an MCB similar to the one $\overline{z}$ has. Afterwards, the remaining m objects' MCBs ($m \leq k$) are used to calculate the ratio of correct decisions for each single classifier, resulting the local explicit confidence.

In this case, similarity has a greater importance than in the feature space, because in the MCB space, the dimensions are far from being normalized. Even if the confidences (LICs) in these dimensions are normalized, it is hard to compare LIC values of different individual classifiers. This means that the same distance would mean a different numeric threshold in the dimensions that those individual classifiers are responsible for. For instance, in DT algorithms, the typical distance between LICs are 10 or 100 times greater, than in NN based classifiers. The solution is to normalize each dimension using the average distance of LICs. This average

is calculated for each classifier D_i, and then for all x_j neighbors the $d_i(\overline{z}) - d_i(\overline{x_j})$ distance gets divided by it.

3.4 Combination of the Three Levels

The implicit level and the two explicit levels are representing classifier accuracy from different aspects, due to the dissimilarity of the classifier's inner process based (implicit) and the statistically measured (explicit) approach. Meanwhile, the two explicit levels are more similar. Beyond the usage of implicit measures, the application of two explicit indices is one of the most important difference between our 3-level technique and dynamic classifier selection of Giacinto et al. [8]. Our aim is to rank the classifier ensemble members as precisely as possible, weight them, and combine those that have the potential to improve the whole ensemble's decision. We do not presume that the best decision at classifying a data point can be made by only one classifier. In our approach, we assume that all members are responsible for the whole feature space, instead of just partitions of it. A well-chosen *selection threshold* is used to decide whether the given classifier should be involved in making the decision for a given data point or not. Thus, it is a fusion-selection type method, having the advantage of being more flexible.

Our task is to assign weights to all ensemble members, so that their weighted and summed decisions result a more accurate final decision. Any of the three confidences can be used as weights. Although, using all three confidence levels, and also in a selection-fusion manner is advisable. This enables the method to adapt at the same time to data points (1) that are very different from the training set, (2) that fall within more than one of the ensemble members' expertise, thus all these members can classify them with good accuracy, and (3) that don't clearly fall within any of the classifiers expertise, based on the locality computations.

For instance, a simple classifier selection chooses the best member of the ensemble to make the decision. Practically it is the member, that classified most data points correctly in the local region. The assumption of (1) results that the data points forming the local region will be highly different from the data point to be classified. Thus, the best member will not be able to classify the data point correctly. This means that the problem of encountering dissimilarities between the data point and its local region escalates also to the classifier selection. All in all, in this case it is highly probable, that a different ensemble member is selected, instead of the best one for the data point to be classified. With the help of our multi level technique, it's possible to detect these situations. If the local region becomes practically empty or highly unreliable because of these dissimilarities, our method uses the GEC instead of the LEC.

The second type of problematic data points are less critical, since in these cases even only one member is able to make the good decision with relatively high confidence. In (2), again the challenge is that the local region can not be determined perfectly. Hence, weights calculated for members of the ensemble, based on this neighborhood does not really guarantee a clear ranking among the several classifiers that are all have good expertise in that local region. Picking one that seems

to be the best can be a defective choice. This is the typical case of classifier fusion overshadowing selection. Considering a subset of generally accurate individual classifiers with divers errors, our t_{sel} selection threshold enables each of these members, due to their high performance, and fusion will filter out errors.

And finally, solving the third problematic case is the hardest using only selection methods. When non of the individual members could make the right decision on their own. Choosing amongst almost equally bad classifiers clearly would not be a good idea. Instead, by identifying those that have a recognizable confidence and combining their weighted decisions, the result can be much more satisfiable. Moreover, the weighting can be further improved by combining the LEC and the GEC, assuming that some accuracy differences between members were masked by local region computation error.

LEC and GEC are utilizing the LIC values, as it was described in Sections 3.2 and 3.3. However, the LIC index can be used also to complete the above described weightings. Practically, we can regard it as the basic probability of the decision's correctness. It is a much more sophisticated measure to use, instead of weighting only the decisions.

Now we demonstrate the three-level voting method by and example. Assuming the ensemble has two members ($L = 2$), a binary classification task ($C = 2$), 10 training data points ($n = 5$), and 1 unlabelled test data point ($\overline{z}$). The members are trained (black-boxes), their LIC values and decisions are given in Table 2 by a combined LIC metric often used at binary classification tasks ($LIC^{comb} \in [0, 1]$). This index describes the decision and its confidence as well. The closer this value is to 0 or 1, the higher the confidence is. Thus a value of 0.5 would mean a random decision with zero confidence. From now on, we use LIC in the example as a shorter name that stands for LIC^{comb}. Then we calculate the GEC for both classifiers ($i = 1, 2$). Let the evaluation method be the number of correct decisions.

$$GEC_{D_i} = \frac{\sum_{j=1}^{n} \frac{sgn(0.5 - \left|LIC_{D_i,\overline{x}_j} - y_j\right|)}{2} + 0.5}{n},$$

assuming that $sgn(0) = 0$.

$$GEC_{D_1} = \frac{3}{5} = 0.6, \tag{4}$$

$$GEC_{D_2} = \frac{2}{5} = 0.4, \tag{5}$$

To calculate the LEC, that also uses LIC, first, we have to gather the neighborhood $X_{nb}(\overline{z})$ of the test point. First, k-NN is calculated from the N features of $\overline{x}_5$ and $\overline{z}$, where k is a parameter of the algorithm (now $k = 3$). The resulting training dataset is given in Table 3. The second step is the MCB filtering of $X_{nb}(\overline{z})$.

$$X_{nb}^{MCB}(\overline{z}) = \left\{\overline{x}_i \in X_{nb}^{k-NN}(\overline{z}) | dist_{MCB}(\overline{x}_i) \leq t_{MCB}\right\},$$

where the t_{MCB} threshold is a parameter of the algorithm. The MCB distance is calculated with the above mentioned MCB normalization method, by the $dens_{D_i}$

average LIC density of classifiers (Section 3.3).

$$dist_{MCB}(\overline{x}_j) = \sqrt{\sum_{i=1}^{L} \left(\frac{LIC_{D_i,\overline{x}_j} - LIC_{D_i,\overline{z}}}{dens_{D_i}} \right)^2}$$

$$dens_{D_i} = \frac{\sum_{j=1}^{n-1} \left| LIC_{D_i,j}^{sorted} - LIC_{D_i,j+1}^{sorted} \right|}{\sum_{j=1}^{n-1} sgn \left| LIC_{D_i,j}^{sorted} - LIC_{D_i,j+1}^{sorted} \right|},$$

assuming that $sgn(0) = 0$.

$$dens_{D_1} = \frac{0.01 + 0.02 + 0.03}{3} = 0.02 \tag{6}$$

$$dens_{D_2} = \frac{0.1 + 0.1 + 0.1 + 0.3}{4} = 0.15 \tag{7}$$

$$dist_{MCB}(\overline{x}_2) = \sqrt{\left(\frac{0.07 - 0.01}{0.02} \right)^2 + \left(\frac{0.5 - 0.45}{0.15} \right)^2} = 3.02 \tag{8}$$

Results of MCB distance calculations for the whole k-NN local region (X_{nb}^{k-NN}) is shown in Table 3. In the example, we choose our second parameter $t_{MCB} = 2.5$, hence $X_{nb}^{MCB} = \{\overline{x}_4, \overline{x}_5\}$.

Table 2: Combined LIC values of the example

X	LIC_{D_1}	LIC_{D_2}	y
$\overline{x}_1$	0.01	0.2	0
$\overline{x}_2$	0.07	0.5	1
$\overline{x}_3$	0.01	0.3	0
$\overline{x}_4$	0.04	0.8	0
$\overline{x}_5$	0.02	0.4	1
$\overline{z}$	0.02	0.4	?

Table 3: k-NN filtered data point's combined LIC values of the example

$X_{nb}^{k-NN}(\overline{z})$	LIC_{D_1}	LIC_{D_2}	y	$dist_{MCB}$
$\overline{x}_2$	0.07	0.5	1	3.02
$\overline{x}_4$	0.04	0.8	0	2.32
$\overline{x}_5$	0.02	0.4	1	2.03
$\overline{z}$	0.01	0.45	?	0

The next step in deriving LEC is to calculate the average error rate of the ensemble members in this local region. The average distance of this error rate and the $LIC = 1$ value yields the confidence index we are looking for (higher confidence at lower error rate).

$$LEC_{D_i,\overline{z}} = \frac{\sum_{\forall j,\overline{x}_j \in X_{nb}^{MCB}} \left(1 - \left|LIC_{D_i,\overline{x}_j} - y_j\right|\right)}{\left|X_{nb}^{MCB}\right|},$$

if $\left|X_{nb}^{MCB}\right| \neq 0$. Generally, we produce a small neighborhood, and thus derive the LEC values quite strictly. Due to this, X_{nb}^{MCB} is a fairly small set, the numerosity is zero quite often. In this case, ensemble members' weight is determined by the GEC as we stated previously.

$$LEC_{D_i,\overline{z}} = GEC_{D_i},$$

if $\left|X_{nb}^{MCB}\right| = 0$. We can see in Equation (9) and (10), that neither of the individual classifier members of the ensemble has a convincing confidence based on the neighborhood.

$$LEC_{D_1,\overline{z}} = \frac{(1 - 0.04) + (1 - 0.98)}{2} = 0.49 \tag{9}$$

$$LEC_{D_2,\overline{z}} = \frac{(1 - 0.8) + (1 - 0.6)}{2} = 0.3 \tag{10}$$

The third parameter of our algorithm is the $t_{sel-fus}$ selection-fusion threshold. It is used to determine the minimum LEC value of D_i members that are selected and considered for fusing. We chosen $t_{sel-fus} = 0.25$ in our example. The resulting $TLV(\overline{z})$ probability that serves as the output of our method is calculated in Equation (11). Naturally, at least one member always has to be collected, otherwise no weight could be assigned. The TLV index of our Three-Level Voting strategy then derived using all three confidence measures (GEC is used implicitly in LEC).

$$TLV(\overline{z}) = \frac{\sum_{\forall i, D_i \in D_{sel}(\overline{z})} (LIC_{D_i,\overline{z}} \cdot LEC_{D_i,\overline{z}})}{\sum_{\forall i, D_i \in D_{sel}(\overline{z})} LEC_{D_i,\overline{z}}}$$

$$D_{sel}(\overline{z}) = \{D_i | LEC_{D_i,\overline{z}} \geq t_{sel-fus}\}$$

$$TLV(\overline{z}) = \frac{0.01 \cdot 0.49 + 0.45 \cdot 0.3}{0.49 + 0.3} = 0.18 \tag{11}$$

Since D_1 is proved to be more accurate than D_2 on neighboring data points, we can see that $LIC_{D_1,\overline{z}}$ is stressed in (11), compared to the LIC of D_2.

4 Results

To measure the three-level confidence voting technique's accuracy and performance, we aimed to use a dataset widely available on the Internet. Furthermore, we wanted

to prove the expected performance in a manner not only reproducible, but immediately comparable as well. Hence, we decided to test the method on the dataset of ACM SIGKDD's KDD Cup 2008, the oldest and most famous among the data mining competitions [1, 2]. Before going into details about the results, we present this dataset [3], provided by Siemens Medical Solutions USA.

4.1 KDD Cup 2008

The competition's main task was to detect breast cancer at an early stage from X-ray images of the breast. This typically consists of 4 X-ray images, 2 images per breast that are taken from different directions, called MLO and CC. The data is preprocessed in 2 steps.

1. Candidate generation from suspicious regions of the X-ray images.
2. Feature extraction, that computes descriptive features for each candidate to represent them in the form of numerical values.

Our task is to classify candidates to be able to differentiate between malignant cancers from the rest of the candidates based on the given feature vector. The rate of prevalence is extremely low, only 0.5-1% of the patients have breast cancer. The entries are judged by the area under the FROC curve, in the clinically relevant region 0.2-0.3 false positives per image. The participants had to return a confidence score for every candidate that indicates the confidence of their classifier that the candidate is malignant.

The training data consists of 102294 candidates from 118 malignant, and 1594 normal patients, each described with 117 features, and also an information segment about it's source image, patient, candidate's coordinates on the image, and a class label indicating whether or not it is malignant. The features are computed from several standard image processing algorithms, but there are no additional proprietary features.

4.2 Comparison of Results

First of all, we're going to introduce the individual classifier that are used to form ensembles and demonstrate their individual performance in the above mentioned classification challenge. 17 classifiers are tested including Neural Network (IDs beginning with 4), Support Vector Machine (IDs beginning with 5), Logistic Regression (IDs beginning with 1 or 3) and Decision Tree (IDs beginning with 2) algorithms. The ID 1 was assigned to the classifier with the best individual performance (later marked by GEC**), thus we can see that it was a logistic regression model. The goodness of the individual classifiers is shown in Table 4.

These confidence values were provided by the KDD Cup's evaluation method, which is based on the area under FROC, that is calculated from the LIC values of the validation partition of the dataset. The method' details are introduced on the web page of the competition [2]. So these individual classifier goodness indices

Table 4: Accuracy (GEC) of individual classifiers in the ensemble

ID(↓)	GEC	ID	GEC(↑)
1	76,79%	1	76,79%
21	55,91%	51	74,55%
22	55,90%	42	72,94%
31	71,98%	31	71,98%
32	63,91%	46	71,93%
33	62,06%	34	71,00%
34	71,00%	47	69,64%
35	69,06%	35	69,06%
41	59,17%	44	68,43%
42	72,94%	52	67,93%
43	56,81%	45	65,77%
44	68,43%	32	63,91%
45	65,77%	33	62,06%
46	71,93%	41	59,17%
47	69,64%	43	56,81%
51	74,55%	21	55,91%
52	67,93%	22	55,90%

are practically the GEC values. First, we are to demonstrate the performance potential in using only the GEC measure for voting, and finally, all three levels are combined and accuracy is tested as the two explicit indices (GEC, LEC) get integrated (Subsection 3.4), still on local implicit basis, forming our Three-Level Voting (TLV) strategy.

Given 17 individual classifiers, this would mean a tremendous amount of possible combinations. Hence, we decided to narrow down the problem space by constructing ensembles with only 2 members (classifier pairs). Evaluating only these pairs resulted 136 possible combinations. Several voting methods were considered, to give a wide range of possible comparison. The 136 ensembles are sorted by best ensemble accuracy reached among tested voting methods. The 5 best performances are listed in Table 5, referred also by ensemble members' IDs.

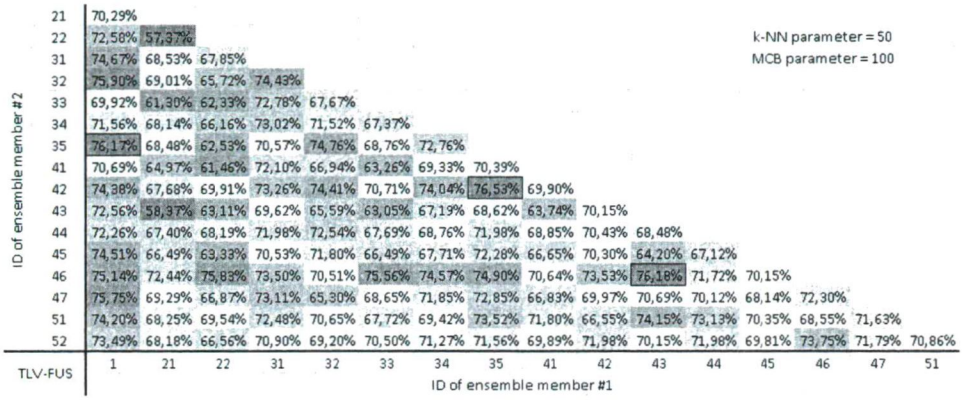
MAJ (majority voting), AVG (average), HC (highest confidence), MAX (maximum), MIN (minimum), GEC-AVG, TLV-SEL, TLV-FUS rows represent the voting methods. Majority voting is the only decision based method amongst the tested. Both classifiers have a vote, equivalent with their decision. In the end, the votes are averaged and the resulting ratio serves as the confidence value required in the evaluation. In the next three methods the combined values can be calculated with the $AVG(x_1, x_2)$, $MIN(x_1, x_2)$, and $MAX(x_1, x_2)$ functions, where x_i represents the LIC values of the two ensemble members.

Table 5: Performance of top 5 ensembles using different voting strategies

Ensemble	#1	#2	#3	#4	#5
ID1	1	1	1	1	1
GEC1	76,79%	76,79%	76,79%	76,79%	76,79%
ID2	35	32	31	42	52
GEC2	69,06%	63,91%	71,98%	72,94%	67,93%
Majority	34,59%	31,79%	38,49%	27,59%	52,32%
Avg	77,53%	76,96%	75,48%	76,38%	72,53%
HC	77,41%	78,88%	78,5%	78,48%	75,35%
Max	76,79%	76,79%	76,79%	76,79%	76,79%
Min	69,06%	65,93%	71,98%	72,95%	71,73%
GEC-AVG	77,53%	77,48%	75,7%	76,38%	72,42%
TLV-FUS	76,17%	75,9%	74,67%	74,38%	73,49%
TLV-SEL	79,63%	76,94%	77,08%	77,3%	78,38%
MaxMeth	2LV-SEL	HC	HC	HC	2LV-SEL
MaxGEC	79,63%	78,88%	78,5%	78,48%	78,38%
DiffMax	2,83%	2,09%	1,71%	1,69%	1,59%
OK	1	1	1	1	1

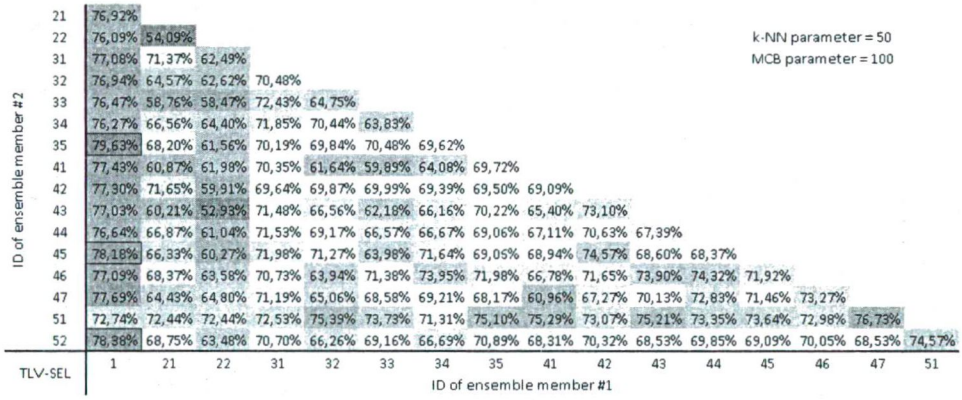
The highest confidence method selects the classifier, which was more confident in it's decision. So the output is the LIC value of the classifier with the highest confidence (the most distant from 0.5, considering *combined LIC*). GEC-AVG is almost the same as AVG, but here, the members are weighted by their GEC index before calculating the average of LIC values. The SEL and FUS versions of the TLV method stand for selection and fusion. Since we are combining classifier pairs, there's only 2 possible implementations of selection-fusion: pure selection (TLV-SEL) and pure fusion (TLV-FUS). Hence, parameter $t_{sel-fus}$ can be omitted. MaxMethod row contains the best method for the given ensemble (column) and MaxGec contains the best method's GEC value. DiffMax shows the precision compared to the best member's GEC value in the ensemble (GEC*). The OK flag value is equal to 1, if MaxGec is equal or higher than the GEC*.

In Table 5, we can see, that the best method amongst all was our TLV-SEL, that reached 79.63%, beating the GEC* by 2.83%. The GEC value of the best classifier amongst our 17 tested is marked by GEC**. Since in this case, the ensemble contained the best classifier of our 17 tested (thus GEC* = GEC**), our method was successful in increasing the overall performance compared to all our individual classifiers. The TLV-SEL algorithm is proved to be the best method also at #5 of the 136 ensembles besides #1, and the second best method at #3 and #4. At #2 it is only the forth, while GEC-AVG was second best, hence it is highly probable that the TLV parameters are far from perfect, otherwise GEC-AVG, that uses GEC, which is also a part of TLV could not be more accurate.



	1	21	22	31	32	33	34	35	41	42	43	44	45	46	47	51
21	70,29%															
22	72,58%	57,37%														
31	74,67%	68,53%	67,85%													
32	75,90%	69,01%	65,72%	74,43%												
33	69,92%	61,30%	62,33%	72,78%	67,67%											
34	71,56%	68,14%	66,16%	73,02%	71,52%	67,37%										
35	76,17%	68,48%	62,53%	70,57%	74,76%	68,76%	72,76%									
41	70,69%	64,97%	61,46%	72,10%	66,94%	63,26%	69,33%	70,39%								
42	74,38%	67,68%	69,91%	73,26%	74,41%	70,71%	74,04%	76,53%	69,90%							
43	72,56%	58,37%	63,11%	69,62%	65,59%	63,05%	67,19%	68,62%	63,74%	70,15%						
44	72,26%	67,40%	68,19%	71,98%	72,54%	67,69%	68,76%	71,98%	68,85%	70,43%	68,48%					
45	74,51%	66,49%	63,33%	70,53%	71,80%	66,49%	67,71%	72,28%	66,65%	70,30%	64,20%	67,12%				
46	75,14%	72,44%	75,83%	73,50%	70,51%	75,56%	74,57%	74,90%	70,64%	73,53%	76,18%	71,72%	70,15%			
47	75,75%	69,29%	66,87%	73,11%	65,30%	68,65%	71,85%	72,85%	66,83%	69,97%	70,69%	70,12%	68,14%	72,30%		
51	74,20%	68,25%	69,54%	72,48%	70,65%	67,72%	69,42%	73,52%	71,80%	66,55%	74,15%	73,13%	70,35%	68,55%	71,63%	
52	73,49%	68,18%	66,56%	70,90%	69,20%	70,50%	71,27%	71,56%	69,89%	71,98%	70,15%	71,98%	69,81%	73,75%	71,79%	70,86%

Figure 1: Performance of method TLV-FUS on 136 ensembles



	1	21	22	31	32	33	34	35	41	42	43	44	45	46	47	51
21	76,92%															
22	76,09%	54,09%														
31	77,08%	71,37%	62,49%													
32	76,94%	64,57%	62,62%	70,48%												
33	76,47%	58,76%	58,47%	72,43%	64,75%											
34	76,27%	66,56%	64,40%	71,85%	70,44%	63,83%										
35	79,63%	68,20%	61,56%	70,19%	69,84%	70,48%	69,62%									
41	77,43%	60,87%	61,98%	70,35%	61,64%	59,89%	64,08%	69,72%								
42	77,30%	71,65%	59,91%	69,64%	69,87%	69,99%	69,39%	69,50%	69,09%							
43	77,03%	60,21%	52,93%	71,48%	66,56%	62,18%	66,16%	70,22%	65,40%	73,10%						
44	76,64%	66,87%	61,04%	71,53%	69,17%	66,57%	66,67%	69,06%	67,11%	70,63%	67,39%					
45	78,18%	66,33%	60,27%	71,98%	71,27%	63,98%	71,64%	69,05%	68,94%	74,57%	68,60%	68,37%				
46	77,09%	68,37%	63,58%	70,73%	63,94%	71,38%	73,95%	71,98%	66,78%	71,65%	73,90%	74,32%	71,92%			
47	77,69%	64,43%	64,80%	71,19%	65,06%	68,58%	69,21%	68,17%	60,96%	67,27%	70,13%	72,83%	71,46%	73,27%		
51	72,74%	72,44%	72,44%	72,53%	75,39%	73,73%	71,31%	75,10%	75,29%	73,07%	75,21%	73,35%	73,64%	72,98%	76,73%	
52	78,38%	68,75%	63,48%	70,70%	66,26%	69,16%	66,69%	70,89%	68,31%	70,32%	68,53%	69,85%	69,09%	70,05%	68,53%	74,57%

Figure 2: Performance of method TLV-SEL on 136 ensembles

The algorithm uses two main parameters, the k index of the k-NN method, and the t_{MCB} threshold. Our technique's accuracy is highly dependent on these parameters. First, we tuned these two parameters using all 136 ensembles. This resulted a local optimum of $t_{MCB} = 100$, $k = 50$ that performed well on all ensembles. We included the results gathered by parameter setting in Table 5 presented previously. Considering computational expenses, we decided to run k-NN in a faster, approximative manner. This is done by searching k nearest neighbors only in a randomly chosen 1% of the training data points (approximatly 1000 data points). Meanwhile we kept all positive training samples in, to improve the rate of prevalence. The results of this setting is presented in Figure 1 for TLV-FUS, and in Figure 2 for TLV-SEL. The accuracy of the ensembles are demonstrated also by the fill color of the cell. The top three ensembles are marked by red bordering. The best ensemble for these settings was the pair of ID1 and ID35.

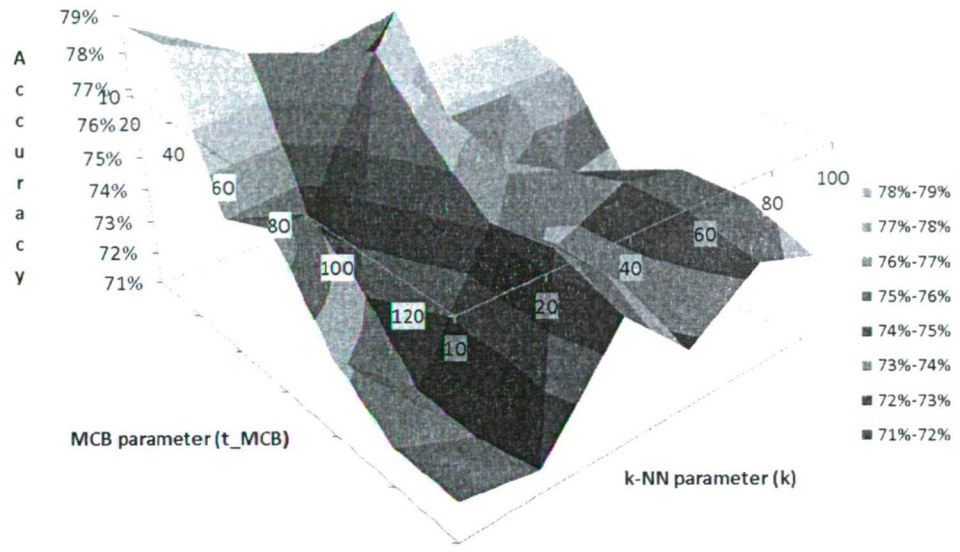


Figure 3: Trends in accuracy of ensemble ID1+ID35, taken as function of parameter settings

TLV-FUS ID1+ID35	k-NN parameter					
MCB parameter	10	20	40	60	80	100
10	78,76%	77,56%	76,34%	74,28%	74,34%	74,34%
20	78,90%	78,09%	76,07%	78,17%	78,51%	75,59%
40	75,01%	74,34%	78,20%	75,72%	75,80%	75,77%
60	75,70%	73,19%	76,51%	75,25%	74,14%	73,47%
80	73,31%	71,28%	75,17%	73,69%	74,72%	74,34%
100	72,12%	71,20%	75,48%	73,59%	74,72%	74,34%
120	72,15%	71,69%	74,75%	72,77%	74,30%	73,59%

Figure 4: Numeric accuracy values of ensemble ID1+ID35, taken as function of parameter settings

Afterwards, we searched the optimal parameter settings considering this pair of classifiers (ensemble ID1+ID35). This time, we applied k-NN on 10% of the training points plus positive samples to increase k-NN precision. Figure 3 demonstrates the trends in accuracy of method TLV-FUS on a 3D surface, while Figure 4 is the same result presented in 2D with numeric values of accuracy assigned to each parameter setting. These are plotted also for method TLV-SEL in Figures 5 and 6. The floor of the 3D figures represents the GEC*. On the 2D figures, red bordering is applied where the accuracy of the ensemble at that parameter setting is greater then the GEC* hence the combination successfully improved performance. Furthermore,

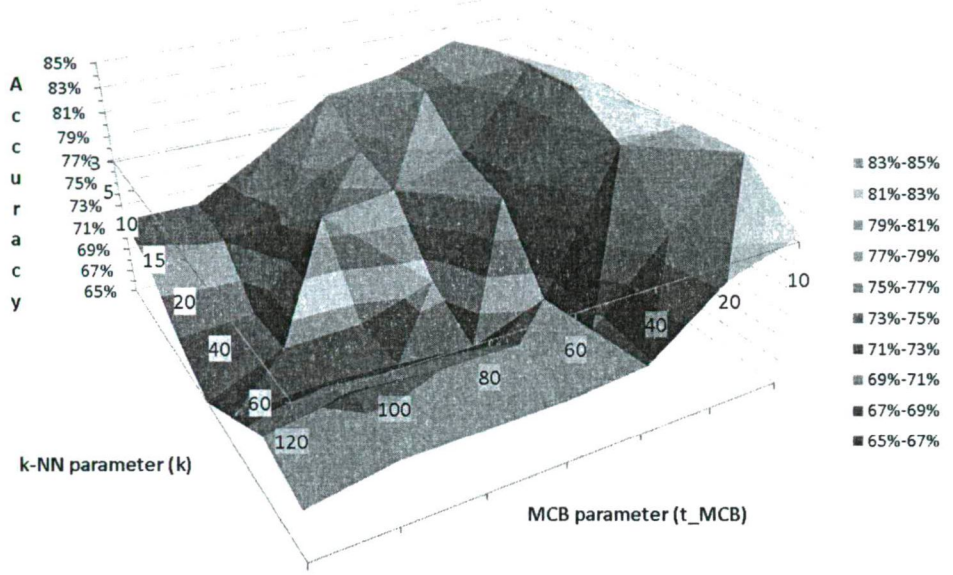


Figure 5: Trends in accuracy of ensemble ID1+ID35, taken as function of parameter settings

TLV-SEL	k-NN parameter						
ID1+ID35	3	5	10	15	20	40	60
MCB parameter 10	79,53%	80,11%	81,13%	80,66%	81,12%	81,13%	77,26%
20	81,77%	82,79%	83,21%	83,21%	80,71%	76,32%	75,45%
40	79,80%	80,14%	76,70%	75,45%	69,54%	70,12%	70,71%
60	78,89%	76,11%	74,97%	69,76%	67,26%	70,95%	70,17%
80	74,96%	73,61%	74,20%	67,84%	67,79%	71,01%	70,17%
100	71,63%	71,14%	67,37%	67,67%	68,20%	71,37%	70,17%
120	71,65%	70,53%	67,06%	64,46%	67,63%	70,31%	69,06%

Figure 6: Numeric accuracy values of ensemble ID1+ID35, taken as function of parameter settings

cells containing the highest accuracy are marked by red text. By method TLV-FUS, this accuracy is 78.9% at parameters ($k = 10; t_{MCB} = 20$), while by using TLV-SEL, it is 83.21% at parameters ($k = 10, 15; t_{MCB} = 20$). The performance improvement compared to GEC* (76.79%) is *2.11%* by TLV-FUS and *6.42%* by TLV-SEL. The method that reached second best ensemble accuracy considering all 136 ensembles was HC with 78.88% in Ensemble #2 (in Ensemble #1 HC only reached 77.41%). TLV-FUS managed to surpass the best performance of HC by 0.02%, while TLV-SEL overshadows it by *4.33%*.

We separated an independent, labeled data partition for final evaluation purposes, to be able to test the real accuracy of the best, optimized ensemble, with tuned parameters. For comparison, we also reevaluated the accuracy of our individ-

Table 6: Accuracy of our technique evaluated on an independent dataset

$t_{sel-fus}$	k	t_{MCB}	GEC^{TLV}	$GECDiff^{**}$	$GECDiff^{HC}$
fusion	10	20	78.91%	0.12%	-1.66%
selection	10	20	84.04%	5.25%	3.47%
selection	15	20	84.36%	5.57%	3.79%

ual classifiers (GEC^{TLV}). The best individual classifier reached a GEC of 78.79% (still Classifier ID1: GEC^{**}). The second best ensemble accuracy reached 80.57% (HC method, Ensemble #2: GEC^{HC}) on this independent partition. While the three best ensembles that use our algorithm among ensembles built of Classifier ID1 and Classifier ID35 are presented in Table 6, detailed with parameter settings, where $GECDiff^{**} = GEC^{TLV} - GEC^{**}$ and $GECDiff^{HC} = GEC^{TLV} - GEC^{HC}$.

5 Conclusion

In out paper, by using the confidence triplet of "general accuracy", "self confidence", and "expertise on the particular field" at classifying data objects, we created a novel meta-classifier stage classifier ensemble algorithm. This fusion-selection method is able to outdo the best of the combined classifiers' accuracy by more than 5% and also overshadows the second best meta-classifier stage algorithm by more than 3%. Hence, we experimentally proved that our algorithm has the potential to significantly increase classification performance compared to individual classifiers and other voting techniques. Considering future work, there are still numerous aspects that can be improved. Since in this paper only one ensemble was subjected to parameter optimization, which was selected by local optimization, it is very probable that a global optimization can reveal even more potential. There is the 3 dimensional space of (1) which individual classifier pairs to include in the ensemble, (2) and (3) setting the two main parameters. Furthermore, considering not only classifier pairs as ensembles, the third parameter of our technique is also present, instead of just dividing the space into the two segments of selection and fusion. Hence (4) setting the selection-fusion threshold. And finally, our algorithm will be tested on other well-known datasets as well.

References

[1] The 14th acm sigkdd international conference on knowledge discovery and data mining. http://www.sigkdd.org/kdd2008/, 2008.

[2] Kdd cup 2008 on mining medical data. http://www.kddcup2008.com/, 2008.

[3] The original data set of kdd cup 2008 competition on mining medical data. http://adatbanyaszat.tmit.bme.hu/ fozocsaba/kdd2008/, 2008.

[4] Breiman, L. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[5] Breiman, L. Arcing classifiers. *Annals of Statistics*, 26(3):801–849, 1998.

[6] Breiman, L. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[7] Freund, Y. and Schapire, R. E. Experiments with a new boosting algorithm. In *Thirteenth International Conference on Machine Learning*, pages 148–156, San Francisco, 1996. Morgan Kaufmann.

[8] Giacinto, Giorgio and Roli, Fabio. Dynamic classifier selection based on multiple classifier behaviour. *Pattern Recognition*, 34:1879–1881, 2001.

[9] Ho, T.K. The random subspace method for constructing decision forests. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 20(8):832–844, Aug. 1998.

[10] Kuncheva, L. I. *Combining Pattern Classifiers. Methods and Algorithms*. John Wiley and Sons, 2004.

[11] Kuncheva, L. I. Diversity in multiple classifier systems (editorial). *Information Fusion*, 6(1):3–4, 2004.

[12] Kuncheva, L. I. and Rodríguez, J. J. Classifier ensembles with a random linear oracle. *IEEE Transactions on Knowledge and Data Engineering*, 19(4):500–508, 2007.

[13] Kuncheva, L. I. and Rodríguez, J. J. An experimental study on rotation forest ensembles. In *7th International Workshop on Multiple Classifier Systems, MCS 2007, volume 4472 of LNCS*, pages 459–468. Springer, 2007.

[14] Long, P.M. and Vega, V.B. Boosting and microarray data. *Machine Learning*, 52:31–44, 2003.

[15] Margineantu, D. D. and Dietterich, T. G. Pruning adaptive boosting. In *Proc. 14th Int'l Conf. Machine Learning*, pages 211–218, 1997.

[16] Melville, P., Shah, N., Mihalkova, L., and Mooney, R.J. Experiments with ensembles with missing and noisy data. In *Proc Fifth Int'l Workshop Multiple Classifier Systems*, pages 293–302, 2004.

[17] Oza, N. C. Boosting with averaged weight vectors. In *Proc Fourth Int'l Workshop Multiple Classifier Systems (MCS 2003)*, 2003.

[18] Piatetsky-Shapiro, Gregory. Kdnuggets poll: Which analysis types you plan to do more in 2008. http://www.kdnuggets.com/polls/, 2007.

[19] Rodríguez, J. J., Kuncheva, L. I., and Alonso, C. J. Rotation forest: A new classifier ensemble method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1619–1630, 2006.

[20] Webb, G. I. Multiboosting: A technique for combining boosting and wagging. *Machine Learning*, 40(2):159–196, 2000.

[21] Y.S. Huang, C.Y. Suen. A method of combining multiple experts for the recognition of unconstrained handwritten numerals. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 17(1):90–94, 1995.

Acta Cybernetica **19** (2009) 61–92.

Clouds, p-boxes, Fuzzy Sets, and Other Uncertainty Representations in Higher Dimensions

Martin Fuchs*

Abstract

Uncertainty modeling in real-life applications comprises some serious problems such as the curse of dimensionality and a lack of sufficient amount of statistical data.

In this paper we give a survey of methods for uncertainty handling and elaborate the latest progress towards real-life applications with respect to the problems that come with it. We compare different methods and highlight their relationships. We introduce intuitively the concept of potential clouds, our latest approach which successfully copes with both higher dimensions and incomplete information.

Keywords: uncertainty models, potential clouds, confidence regions, higher dimensions, incomplete information, reliability methods, p-boxes, Dempster-Shafer theory, fuzzy sets

1 Introduction

Among the major problems in real-life applications of uncertainty representations we have identified two particularly complicated ones. One concerns the dimensionality issue. High-dimensionality can cause computations to become very expensive, with an effort growing exponentially with the dimension in many cases. This phenomenon is famous as the curse of dimensionality [45]. Even given the full knowledge about a joint distribution the numerical computation of error probabilities may be very time consuming, if not impossible. Moreover, rigorous computation or (preferably tight) bounding of failure probabilities can only be done in very few cases because the space of possible scenarios is too large. In higher dimensions full probabilistic models need to estimate high-dimensional distributions for which rarely sufficient data are available. Frequently it is just the other way around, i.e., statistical data are scarce. This leads to the second issue which is incomplete, imprecise, or subjective information. Thus we can formulate our ultimate question for discussing an uncertainty method: How does the quality of the method respond

*University of Vienna, Faculty of Mathematics, Nordbergstr. 15, 1090 Wien, Austria, E-mail: martin.fuchs@univie.ac.at, www.martin-fuchs.net

to a **lack of information** and to the **dimensionality** of the problem to solve? The dimension of a problem is determined by the number of uncertain variables involved. In some real-life design problems the dimension is low (say, smaller than about 5). In many problems, however, the dimension is significantly higher.

We will see that many methods exist for uncertainty modeling. Depending on the uncertainty information available one identifies the problem class and applies a suitable method for that class. It is just like choosing the appropriate tool from a toolbox. This point of view can be described as a **toolbox philosophy** [74]. This strategy to solve a problem is defined by the problem itself and the characteristics of the uncertainties involved. Thus different approaches to uncertainty modeling do not contradict each other, but rather constitute a mutually complementing framework.

The most general point of view describing scarce, vague, incomplete, or conflicting uncertainty information are **imprecise probabilities** [95]. This approach alludes to existing uncertainty models being not sufficiently general to handle all kinds of uncertainty, and it encourages to develop a **unified formulation** [97], opposed to the toolbox philosophy. Thus it rather aims at unification on a theoretical basis, whereas our focus is on applicability in real-life reliable engineering.

Uncertainty models in engineering applications are typically employed in the context of design optimization. An uncertainty method should enable to conduct a safety analysis for a given design and to weave this analysis into an optimization problem formulation as safety constraints towards finding a robust, optimal design point.

This paper presents a survey of conventional and modern approaches to uncertainty handling. For each method, the notation, the type of input information, and the basic concepts will be introduced. We will discuss the necessary assumptions, the rigor of results, and the sensitivity of the results to a lack of information. Typically, the more general a method is, the more expensive it becomes computationally, so we will also comment on computational effort, especially in higher dimensions. Eventually, we will highlight relationships between the presented methods and possible embedding in design optimization problems.

We start with a section on the basic principles used in uncertainty handling. Then we present the several different approaches to uncertainty handling: reliability methods, p-boxes, Bayesian methods, Dempster-Shafer theory, fuzzy sets, convex methods, and potential clouds. By means of the potential clouds formalism we present a reliable and tractable worst-case analysis for the given high-dimensional information. This approach enables to determine a nested collection of confidence regions parameterized by the confidence level α, and has already been successfully applied to real-life engineering problems [31], [73] in 34 and 24 dimensions, respectively.

2 Basic principles

Throughout this study we assume familiarity with the basic principles of probability theory. In this section we introduce the notation and fundamental concepts which are the basis for classical methods of uncertainty modeling and of many modern methods as well.

We denote the **sample space** by Ω, an n-dimensional **random vector** by $\varepsilon : \Omega \to \mathbb{R}^n$. A **random variable** is a 1-dimensional random vector which we denote by X. We denote the **probability** of the statement given as argument by Pr. We denote the **expectation** of a random vector ε by $\langle \varepsilon \rangle$. We abbreviate the terms **probability density function** by PDF, and **cumulative distribution function** by CDF, respectively.

2.1 Reliability and failure

To employ uncertainty methods in design safety problems, we need to define failure probabilities p_f and reliability R. The **failure probability** of a fixed design is the probability that the random vector ε lies in a set $\mathbb{F}$ of scenarios which lead to design failure. The **reliability** is the probability that the design will perform satisfactorily, i.e.,

$$R := \Pr(\varepsilon \notin \mathbb{F}) = 1 - \Pr(\varepsilon \in \mathbb{F}) = 1 - p_f, \tag{1}$$

so determining R and p_f are equivalent problems. A third important notion is that of a confidence region for ε. A set C_α is a **confidence region** for the confidence level α if

$$\Pr(\varepsilon \in C_\alpha) = \alpha. \tag{2}$$

The relation between confidence regions and failure probabilities can be seen as follows. Assume that we have determined a confidence region C_α for the random vector ε, and C_α does not contain a scenario which leads to design failure, i.e., $C_\alpha \cap \mathbb{F} = \emptyset$. Then $\Pr(C_\alpha \cup \mathbb{F}) = \Pr(C_\alpha) + \Pr(\mathbb{F}) \leq 1$. Hence $p_f = \Pr(\mathbb{F}) \leq 1 - \Pr(C_\alpha) = 1 - \alpha$, the failure probability is at most $1 - \alpha$. For the reliability $R = 1 - p_f$ we get $R \geq \alpha$.

2.2 Incomplete Information

To make use of probabilistic concepts one often assumes that for the random vector ε involved the joint distribution F is precisely known, provided by an objective source of information. In many design problems, the sources of information are merely subjective, provided by expert knowledge. Additionally, in higher dimensions joint distributions are rarely available, and the typically available distribution information consists of certain marginal distributions.

Often one simply fixes the CDF as normally distributed, arguing with the central limit theorem: a sufficiently large amount of statistical sample data justifies the

normal distribution assumption. The critical question is, what is sufficiently large in higher dimensions? The generalized Chebyshev inequality (3) gives rigorous bounds for the failure probability $p_f = \Pr(\mathbb{F})$, in case that $\mathbb{F}(r) = \{\varepsilon \mid ||\varepsilon||_2 \geq r\}$, r a constant radius. If the components of $\varepsilon = (\varepsilon^1, \dots, \varepsilon^n)$ are uncorrelated, have mean 0 and variance 1, we get from [72]

$$p_f = \Pr(\mathbb{F}) \leq \min(1, \frac{n}{r^2}), \tag{3}$$

and this bound can be attained.

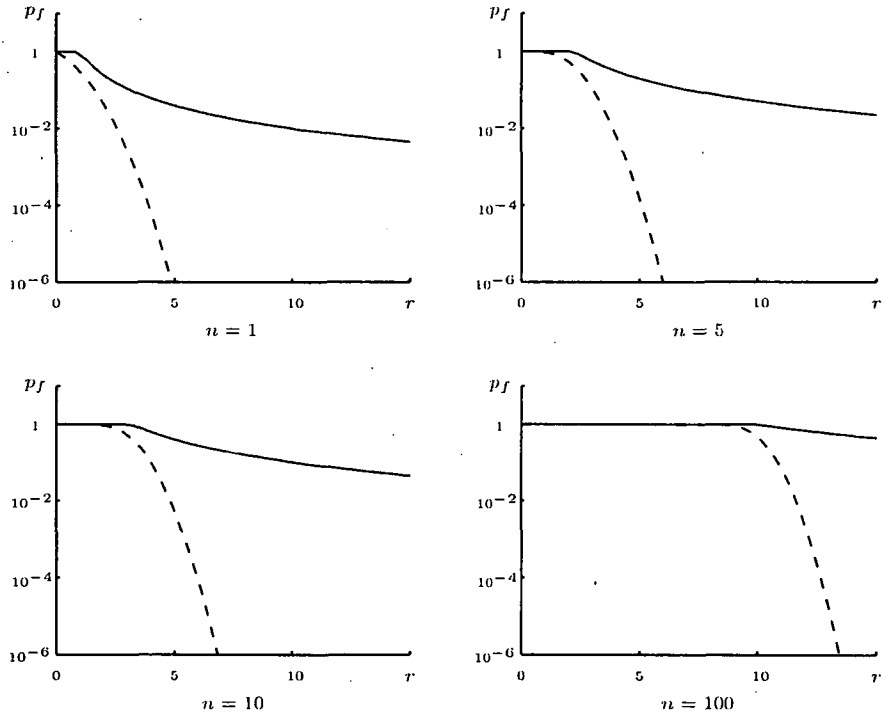


Figure 1: Failure probability p_f for the failure set $\mathbb{F}(r)$ in different dimensions n, bounded from above by the Chebyshev inequality (solid line) and computed from a normal distribution (dashed line), respectively.

The failure probability bounds from (3) differ significantly from those of a normal distribution as shown in Figure 1. If we assume a multivariate normal distribution for ε, uncorrelated is equivalent to independent. The bounds for normal distribution assumption can then be computed from the $\chi^2(n)$ distribution (i.e., a χ^2 distribution with n degrees of freedom). We see that the normal distribution assumption can be much too optimistic compared with the optimal worst-case bounds from (3).

An alternative justification of the normal distribution assumption is the **maximum entropy principle**, if the available information consists of mean and standard deviation only. The principle of maximum entropy originates from information theory [87], and is utilized in many fields of applications, cf., e.g., [34].

The intuitive meaning of entropy is: the larger the entropy the less information (relative to the uniformly distributed improper prior) is reflected in the probability measure with density ρ. In order to define a probability measure given incomplete information, the principle of maximum entropy consists in maximizing the entropy subject to constraints imposed by the information available. For example, in the case of given mean and standard deviation this ansatz leads to a normal distribution, in case of given interval information it leads to a uniform distribution assumption. Note that as soon as we employ the maximum entropy distribution as a probability measure we pretend to have more information than actually available. Hence critical underestimation of failure probabilities may show up.

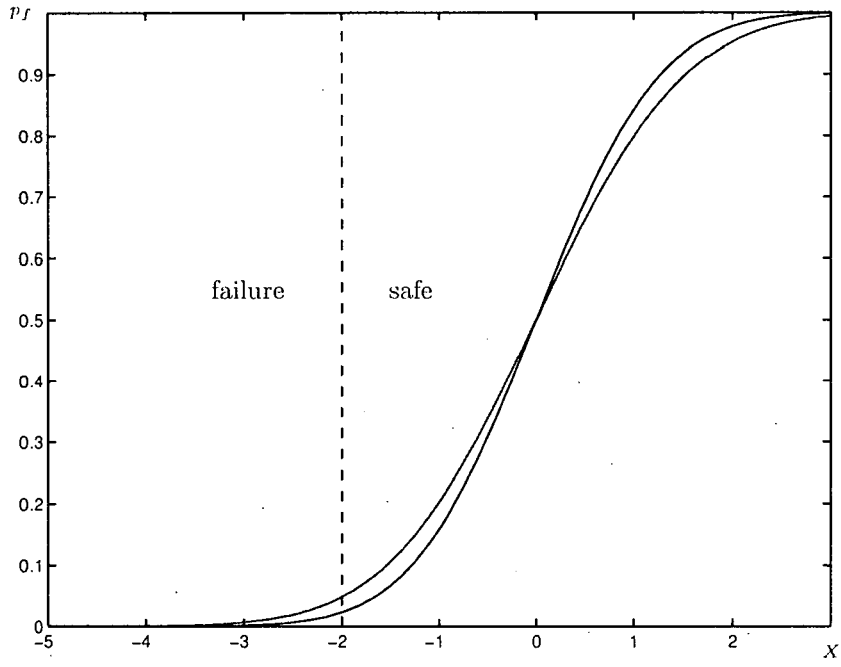


Figure 2: Small deviation in a distribution parameter (here 20% difference in the standard deviation of two normal distributions) can lead to critical underestimation of p_f for a random variable X. Here p_f is underestimated by the factor 2, if the design failure set was $\mathbb{F} = \{X \mid X \leq -2\}$.

In a nutshell, the concept of random variables and probability spaces enables one to derive rigorous statements about failure probabilities and reliability. But they require the probability measure to be precisely known. Otherwise, tails of CDFs

can be critically underestimated, so the estimation of failure probabilities becomes quite poor. In [18] one can find a demonstration that straightforward probabilistic computations are highly sensitive to imprecise information. Imagine a CDF is known almost precisely, but with a small deviation in some distribution parameter. This may easily lead to a situation as shown in Figure 2, which illustrates the fact that the failure probability is often underestimated (here by the factor 2).

In the univariate case it is simple to overcome problems with lack of information. One can apply Kolmogorov-Smirnov (KS) statistics as a powerful tool. Assume that the uncertainty information is given by empirical data on a random variable X, e.g., a small set of sample points $x_1, \ldots, x_N$. The **empirical distribution** $\widetilde{F}$ is defined by

$$\widetilde{F}(\xi) := \sum_{\{j | x_j \le \xi\}} \frac{1}{N}. \tag{4}$$

The KS test uses $D := \max |\widetilde{F} - F|$, the maximum deviation of $\widetilde{F}$ from F, as its test statistics, and it can be shown that $\sqrt{N}D$ converges in distribution to the Kolmogorov function

$$\phi(\lambda) := \sum_{k=-\infty}^{+\infty} (-1)^k e^{-2k^2\lambda^2} \tag{5}$$

for $N \to \infty$, cf. [46]. Conversely, if we choose a fixed confidence level α, we can compute D from

$$D = \frac{\phi^{-1}(\alpha)}{\sqrt{N} + 0.12 + \frac{0.11}{\sqrt{N}}}, \tag{6}$$

cf. [81], and thus find a maximum deviation of the unknown F from the known $\widetilde{F}$. That means that we have non-parametric bounds $[\widetilde{F} - D, \widetilde{F} + D]$ enclosing F with confidence α, only given the knowledge of $x_1, \ldots, x_N$.

In case of high-dimensional random vectors, classical probability theory has no means to cope with scarce data as in the univariate case with KS bounds. Although multivariate CDFs can be defined as in the 1D case using the componentwise partial order in $\mathbb{R}^n$, the computational effort for obtaining higher dimensional PDFs and their numerical integration prohibit the reliable use of standard probabilistic methods in higher dimensions.

2.3 Safety margins

A simple and widely spread non-probabilistic uncertainty model is based on so-called **safety margins**. This model is applied when very little information is available, in situations where most information is provided as interval information.

There are different kinds of sources for interval information, e.g., measurement accuracy. Safety margins are a special kind of interval information, namely one

which is provided subjectively by an expert designer, as typically the case in early design phases. If additional information is available, like marginal distributions or safety margins from further experts, the safety margins approach cannot handle it and thus loses some valuable information. Since safety margins are highly subjective information one cannot expect rigorous results for the safety analysis. However, engineers hope to achieve reasonably conservative bounds by using conservative margins.

The first approach – a tool to handle all kinds of interval information – is **interval analysis**, cf. [64], [70]. We write $X \in [a, b]$ for $a \leq X \leq b$ in the univariate case; in the higher dimensional case $\varepsilon = (\varepsilon^1, \ldots, \varepsilon^n)$, we define interval information $\varepsilon \in [a, b]$ pointwise via $a^1 \leq \varepsilon^1 \leq b^1, \ldots, a^n \leq \varepsilon^n \leq b^n$, and call $[a, b]$ a **box**. We always interpret equalities and inequalities of vectors componentwise. In the following we present two frequent approaches to handle the incoming interval information.

Assume that the cost or gain (or another assessment) function $s : \mathbb{M} \subseteq \mathbb{R}^n \to \mathbb{R}^m$, with design space $\mathbb{M}$, models the response function of the design, and the information about the uncertain input vector ε is given by the bounds $\varepsilon \in [a, b] \subseteq \mathbb{M}$. By way of interval calculations one achieves bounds componentwise on $s(\varepsilon)$ – also called an **interval extension** of s.

Computing an interval extension is often affected by overestimation. A variable $\varepsilon^i \in [a^i, b^i]$ should take the same value from the interval $[a^i, b^i]$ each time it occurs in an expression in the computation of s. However, this is not considered by straightforward interval calculations, so the range is computed as if each time the variable ε^i occurs it can take a different value from $[a^i, b^i]$, leading to an enclosure which may be much wider than the range for $f(\varepsilon)$. One possible way out is based on Taylor series, cf. [55]. Nonlinear interval computations in higher dimensions may become expensive, growing exponentially with n, but can often be done efficiently and complemented by simulation techniques or sensitivity analysis, as we will see later. Note that in case that s is given as a black box evaluation routine – as in many real-life applications – the interval extension cannot be determined rigorously anyway. Also interval methods are often not applicable as a toolbox, but require problem specific expert knowledge to overcome overestimation issues.

In the literature we find much utilization of interval computations in uncertainty modeling. Analyzing the statistics for interval valued samples one seeks to bound mean or variance, which are then also interval valued, cf. [49]: Finding an upper bound on the variance is NP-hard, a lower bound can be found in quadratic time. The field of applications of interval uncertainty for uncertainty handling is vast, e.g., [23], [53], [67], [68], [79].

Also probability theory proper makes use of non-probabilistic interval uncertainty models. For example, consider a Markov chain model with transition matrix (p_{ij}), where the transition probabilities p_{ij} are uncertain, and only given as intervals. Then one can build a generalized Markov chain model, cf. [88], [89]. In [30] one can find a study of imprecise transition probabilities in Markov decision processes.

The second approach to handle safety margin interval information is a simpli-

fication of the information by **fixing each uncertain variable** $\varepsilon^i \in [a^i, b^i]$ at the value of one of the safety margins a^i or b^i, and simply insert this value, for instance a^i for all i, as worst-case scenario to compute the worst-case design response $s(a)$. The decision where to fix the worst-case scenario is taken merely subjectively or via a list of relevant cases. A designer may overestimate intentionally the subjective safety margin assignments, e.g., by adding $20\% = 2 \cdot 0.1$ to the nominal interval bounds for a variable, i.e., $\varepsilon \in [a - 0.1(b-a), b + 0.1(b-a)]$, in order to be suitably conservative in computing the worst-case design response.

The computational effort with this latter approach is not very high and also applies well in higher dimensions. Actually, there is no extra effort in addition to the cost for evaluating s involved in this simple uncertainty model.

A field where safety margins are very popular is multidisciplinary design optimization [1]. In many cases, in particular in early design phases, it is common engineering practice to combine the assignment of safety margins with an iterative process of refining or coarsening the margins, while converging to a robust optimal design. The refinement of the intervals is done by experts who assess whether the worst-case scenario determined for the design at the current stage of the iteration process is too pessimistic or too optimistic. The goal of the whole iteration includes both optimization of the design and safeguarding against uncertainties. The achieved design is thus supposed to be robust. This procedure enables a very dynamic design process and quick interaction between the involved disciplines.

All in all, safety margins allow for a simple, efficient handling of uncertainties, also in large-scale problems, if no information is available but an interval bounding from a single source. Otherwise, we have to look for improved methods, which can handle more uncertainty information. It should be remarked that in most cases, even in early design stages, there is more information available than assumed for the safety margin approach.

2.4 Safety factors

Remember the concept of failure probabilities as introduced in Section 2.1. The failure probability was defined as $p_f = \Pr(\mathbb{F})$, where $\mathbb{F}$ was the set of events which lead to design failure. Let $s : \mathbb{R}^n \to \mathbb{R}$ be the design response of a fixed design for uncertain inputs $\varepsilon \in \mathbb{R}^n$. Assume that there is a limit state ℓ for which $s(\varepsilon) < \ell$ means design failure, and $s(\varepsilon) \geq \ell$ represents satisfying design performance, i.e., that $\mathbb{F}$ is defined by

$$\mathbb{F} = \{\varepsilon \mid s(\varepsilon) < \ell\}. \tag{7}$$

The idea behind safety factors is to build the design in a way that the expected value of $s(\varepsilon)$ is greater than the limit state $\ell > 0$ multiplied by a factor $k_{\text{safety}} > 1$, called the **safety factor**. In other words, a design should fulfill

$$\langle s(\varepsilon) \rangle \geq k_{\text{safety}} \ell. \tag{8}$$

where the expectation has to be suitably estimated. For $s(\varepsilon) \in \mathbb{R}^m$ we interpret this definition componentwise for each safety requirement on the design responses $s^1, \ldots, s^m$, and $\ell = (\ell^1, \ldots, \ell^m)$.

Conversely, suppose that we are given a fixed design and $\langle s(\varepsilon) \rangle \geq \ell$, $s(\varepsilon) \in \mathbb{R}$, $s \in C^1$. Define the maximal feasible safety factor as $k := \frac{\langle s(\varepsilon) \rangle}{\ell}$. To see the relation between k and the design failure probability p_f we assume that we have fixed an admissible p_f for the fixed design, then $p_f = \Pr(s(\varepsilon) \leq \ell) = F_s(\ell)$. Here F_s is the CDF of $s(\varepsilon)$ with density ρ_s given by

$$\rho_s(x) = \rho(s^{-1}(x)) \cdot |\det s'(x)|^{-1}, \tag{9}$$

with $|\det s'(x)|$ the absolute value of the determinant of the Jacobian of s. Hence we get $\ell = F_s^{-1}(p_f)$, assuming that F_s is invertible, and

$$k = \frac{\langle s(\varepsilon) \rangle}{F_s^{-1}(p_f)}. \tag{10}$$

As we are applying standard methods from probability theory to compute safety factors, precise knowledge of ρ and of the **limit state function** $s(\varepsilon) - \ell$ is required to achieve rigorous probability statements.

In the lower dimensional case, if ρ is unknown, but a narrow bounding interval and certain expectations (e.g., means and covariances) for the random vector ε are known, safety factors can still be well described approximately. The expectation of smooth functions s of ε is then achievable from the Taylor series for s, cf., e.g., [5], since expectation is a linear operator on random variables – similar to Taylor models for interval computations. The problems mentioned in Section 2.2 concerning lack of information in the higher dimensional case remain.

Probabilistic computation of safety factors is not as much affected by subjective opinions of the designer as, for example, safety margins. Safety factors are directly associated with required reliability. One important subjective decision is how to fix the required reliability or the admissible failure probability, respectively. The decision can be based, e.g., on the assessment of failure cost or on regulations by legislation.

2.5 Simulation methods

Simulation methods are ubiquitous tools, and uncertainty handling is one of their application fields. Simulation means computational generation of sample points as realizations of a random vector ε, assuming that the joint CDF, marginal CDFs, or interval bounds are given. Thus the not necessarily probabilistic uncertainties involved are simulated, which gives rise to the term simulation methods. Simulation methods are also referred to via the terms **random sampling** or **Monte Carlo sampling**.

After sample generation, the design response $s : \mathbb{M} \to \mathbb{R}^m$ is evaluated for each generated sample point. If all or at least a reasonable majority of the points meet the safety requirements $s(\varepsilon) \geq \ell$, the design is considered to be safe.

The core part of simulation is the sample generation. There is a large number of different techniques addressing it. Three classical variants are based on CDF inversion, Markov chains, and Latin hypercube sampling, respectively. CDF inversion requires the CDF to be invertible, and is particularly not applicable in higher dimensions. The Markov chain method constructs a reversible Markov chain using the Metropolis approach [58], or the more general Hastings method [36], by way of a rejection method. The rejection rule assures that it is not necessary to compute p_i (i.e., the stationary probability of the state i), but only the easy-to-compute quotient $\frac{p_j}{p_i}$ of two different states which is independent of the dimension of X_n. This makes the method highly attractive in higher dimensions. The **Latin hypercube sampling** (LHS) method [57], first determines a finite grid of size N^n, where N is the desired number of sample points and n is the dimension of the random vector for which we want to generate a sample. The grid is preferably constructed such that the intervals between adjacent marginal grid points have the same marginal probability. The N sample points $x_1, x_2, \ldots, x_N$, $x_i = (x_i^1, \ldots, x_i^n) \in \mathbb{R}^n$ are then placed to satisfy the Latin hypercube requirement,

$$\text{For } i,k \in \{1,\ldots,N\},\ j \in \{1,\ldots,n\} :\ x_i^j \neq x_k^j \text{ if } k \neq i. \tag{11}$$

This procedure introduces some preference for a simple structure, i.e., we disregard correlations, tacitly assuming independence. The advantage of the method is that the full range of ε is much better covered than with a Markov chain based method, giving deeper insight to the distribution tails of ε. Hence failure probabilities can be better estimated. Moreover, one does not require more sample points for higher n, so the application of LHS in higher dimensions is still attractive.

Often **importance sampling** is used to speed up simulation techniques by a reduction of the number of required simulations, e.g., [37], [44], [92]. The sample points are generated from a different distribution than the actual distribution of the involved random variables. The sampling density is weighted by an importance density, e.g., a normal distribution with standard deviation σ depending on where the most probable failure points are expected, for instance, depending on the curvature of s. Thus the generated sample is more likely to cover the 'important' regions for the safety analysis.

Considering the rigor of the results one should be aware of the fact that no estimation of failure probabilities computed from a simulation technique is a rigorous bound. These methods are based on the law of large numbers, and their results are only valid for a sufficient amount of sample points. It is difficult to assess what 'sufficient amount' means in a higher dimensional space; one might need to generate an excessively large number of sample points for estimating very small failure probabilities. That is why simulation methods are endangered to critically underestimate CDF tails [20]. It gets particularly dangerous when the CDFs to sample from are unknown.

On the other hand, simulation methods are computationally very efficient, they can be parallelized [51], and also apply well in higher dimensions, where almost no alternatives exist at present.

Another important aspect comes with **black box response functions** s. They principally impose no additional difficulties applying simulation methods. However, if the computational cost for evaluating s is very high, problems will arise as simulation typically requires many evaluations, hence is limited to simple models for s, often surrogate functions (cf., e.g., [17], [39]) for more complex models.

As mentioned earlier simulation techniques have many applications, e.g., the computation of multi-dimensional integrals. They are related to many uncertainty methods, also non-probabilistic ones like interval uncertainty.

2.6 Sensitivity Analysis

Sensitivity analysis is actually not an independent uncertainty method itself, it rather applies in several different fields one of which is uncertainty handling. Sensitivity analysis investigates the variability of a model function output $f(\varepsilon)$, $f : \mathbb{R}^n \to \mathbb{R}$, $\varepsilon = (\varepsilon^1, \varepsilon^2, \ldots, \varepsilon^n)^T$, with respect to changes in the input variables ε^i.

To this end one can follow different approaches, e.g., investigate the partial derivatives of f if they are available, using $\frac{\partial f}{\partial \varepsilon^i}$ as an indicator for the influence of ε^i on f. One can also vary a subset of all single input variables ε^i of f while keeping all other inputs constant. Then one assesses the impact of this subset by the variability of the output f by means of some uncertainty methods introduced in this paper, e.g., fuzzy set or simulation methods. Thus one hopes to achieve a dimensionality reduction of f fixing those input variables which turn out to have little influence on f. Frequently, e.g., [50], one assumes monotonicity of f and interval uncertainty of ε, since this enables the use of very fast techniques in higher dimensions, the effort is then growing only linearly in the dimension n.

As a particular case of handling interval uncertainties in high dimensions with computationally expensive black box response functions s we mention the Cauchy distribution based simulation for interval uncertainty [51]: Assuming that the intervals are reasonably small, e.g., given as measurement errors, it is reasonable to assume that s is linear. Generate N independent sample points for the measurement errors from the scaled Cauchy$(0, 1)$ distribution, which is easy as the inverse CDF of a Cauchy$(0, 1)$ distribution is known explicitly in this case. Linear functions of Cauchy distributed variables are again Cauchy distributed [98], with an unknown parameter Θ. Having estimated the parameter Θ, e.g., by means of a maximum likelihood estimator, one can infer probabilistic statements about errors in s which are Cauchy$(0, \Theta)$ distributed. Thus this method exploits the characteristics of a Cauchy distribution to produce results the accuracy of which can be investigated statistically depending on N, also for low N in case of expensive s. No derivatives are required, only N black box evaluations of s.

Applications of sensitivity analysis can be found, e.g., in [77], [80].

3 Reliability methods

Reliability methods are a very popular approach based on the concepts of reliability and failure probability and transformation to standard normal space, cf. [83]. They represent a significant improvement in computational modeling of reliability compared to rather old-fashioned methods like safety factors.

In order to investigate the failure probability p_f given the joint CDF F, or at least marginal CDFs, of the involved random vector ε, one first applies a coordinate transformation $u = \mathcal{T}(\varepsilon)$ to standard normal space, cf. [69], [84]. Then the failure surface $\mathbb{F} = \{s(\varepsilon) < \ell\}$ is approximated and embedded in an optimization problem to estimate p_f.

Once a $\mathcal{T}$ is found the new coordinates u live in standard normal space, that means the level sets of the density of u are $\{u \mid ||u||_2 = \text{const}\}$, due to the shape of the multivariate normal distribution. Let $s(u)$ be the design response in the transformed coordinates. Then the most probable failure point u^* from the failure set $\mathbb{F} = \{u \mid s(u) < \ell\}$ is the solution of

$$\begin{aligned} &\min_u \ ||u||_2 \\ &\text{s.t.} \ \ s(u) < \ell \end{aligned} \tag{12}$$

i.e., the point from $\mathbb{F}$ with minimal 2-Norm. This critical point is called β-point, and

$$p_f \approx \Phi(-\beta) \tag{13}$$

approximates the failure probability, where $\beta = ||u^*||_2$ and Φ denotes the CDF of the univariate $N(0,1)$ distribution.

Thus we have reduced the estimation of p_f to the standard problem of finding $\mathcal{T}$ and the remaining problem of solving the optimization problem (12). The latter is a nonlinear optimization problem with all the problems that come with it. Even if the limit state function is convex, after transformation it may become a strongly non convex problem in case that the CDF F significantly differs from a normal distribution. Using a linear approximation of the limit state function in the computation of β is called **first order reliability method** (FORM), a quadratic approximation is called **second order reliability method** (SORM).

One hopes that a unique solution for β exists; however in general, there is usually no guaranteed global and unique solution for this optimization problem. Another problem about this approach is that the β-point found may not be representative for the failure probability. A discussion on the involved optimization problem can be studied in [14], investigating difficulties like multiple β-points. The entailed difficulties require some caveats assessing the results of reliability methods: the methods may fail to estimate p_f correctly without warning the user. Especially when additional problems appear – like higher dimensionality or black box response functions s – the reliability methods become less attractive in many large-scale real-life situations. It should be remarked that the search for u^* can be supported by

sampling and simulation techniques like importance sampling, cf. Section 2.5, as means for corrections and reduction of the computational effort, e.g., [56].

Reliability methods are associated with design optimization within the field of reliability based design optimization (RBDO). Instead of the often occurring bilevel problem formulation (i.e., design optimization in the outer level, worst-case scenario search in the inner level) one formulates a one level problem as follows, cf. [43], [65]. Let $s_T = s_T(\theta, u) = s(\theta, T(\varepsilon))$, the design response in transformed coordinates, with the controllable design vector θ which fully specifies the design. Let $g(\theta)$ be the objective function, e.g., the cost of the design or the cost of failure. One seeks to minimize g subject to some reliability constraint $p_f \leq p_a$ where $p_f = \Pr(s(\theta, u) < \ell)$ is approximated by equation (13), and p_a the fixed admissible failure probability. We get

$$\begin{aligned} \min_{\theta} \quad & g(\theta) \\ \text{s.t.} \quad & \Phi(-\beta) \leq p_a \\ & \theta \in T \end{aligned} \tag{14}$$

where T is the set of possible design choices. For $s(\varepsilon) \in \mathbb{R}^m$ we have several constraints $\Phi(-\beta_i) \leq p_a$, $i = 1, \ldots, m$.

Usually simulation techniques are employed to solve (14), e.g., [85]. In [78] it is suggested to use Monte Carlo methods to check the probabilistic constraints, and to train a neural network to check the deterministic constraints, or even both probabilistic and deterministic. This can be implemented as parallelized computations which improves computation time significantly. In any case, one should be aware that one uses a soft solution technique on top of a soft uncertainty model.

4 p-boxes

A p-box – or p-bound, or probability bound – is an enclosure of the CDF of a univariate random variable X, $F_l \leq F \leq F_u$, in case of partial ignorance about specifications of F. Such an enclosure enables, e.g., to compute lower and upper bounds on expectation values or failure probabilities.

There are different ways to construct a p-box depending on the available information about X, cf. [22]. For example, assume that we have empirical data for X. Then we can construct a p-box with KS statistics, cf. (6), after fixing a confidence level α. In [25] we find an exhaustive description which construction techniques can be applied to construct a p-box, related to the type of available information. Moreover, it is illustrated how to construct p-boxes from different uncertainty models like Dempster-Shafer structures (cf. Section 6) or Bayesian update estimates (cf. Section 5). The studies on p-boxes have already lead to successful software implementations, cf. [6], [21].

Higher order moment information on X (e.g., correlation bounds) cannot be handled or processed yet. This is a current research field, cf., e.g., [24].

To compute functions f of p-boxes, that means we have a p-box for $\varepsilon^1, \ldots, \varepsilon^n$ and seek a p-box for $f = f(\varepsilon^1, \ldots, \varepsilon^n)$, one first regards f consisting of elementary arithmetical operations and finds bounds for these expressions. To this end one discretizes the bounds for $\varepsilon^1, \ldots, \varepsilon^n$ towards a discretization of the bounds for f, and then finds an expression for the bound of f in terms of the bounds for $\varepsilon^1, \ldots, \varepsilon^n$. This can be done for all elementary arithmetic operations, without independence assumption for $\varepsilon^1, \ldots, \varepsilon^n$, cf. [99], [100]. Thus the research on arithmetics for random variables actually builds the foundation of p-boxes. The dependency problem is not trivial, assume that one has independent random variables X, Y, Z, then the variables $S = X + Y$ and $T = Y \cdot Z$ are not independent in general.

One learns that the problem of rigorously quantifying probabilities given incomplete information – as done with probability arithmetic and p-boxes – is highly complex, even for simple problems, e.g., [52]. Due to their constructions the methods are rather restricted to lower dimensions and non-complex models f. Black box functions f cannot be handled as one requires knowledge about the involved arithmetic operations. All in all, they often appear not to be reasonably applicable in many real-life situations. On the other hand, as soon as we can apply methods like p-boxes to calculate with bounds on probability distributions, we are not restricted to the use of selecting less rigorous single distribution assumptions (e.g., maximum entropy) anymore.

Two more remarks about p-boxes: First, the definition of p-boxes can be generalized to higher dimensions based on the definition of higher dimensional CDFs, cf. [15]. However, this has not lead to practical results yet. Second, probability arithmetic can be regarded as a generalization of interval arithmetic which would be the special case given only the information $X \in [a, b]$. It is also related to the world of imprecise probabilities via sets of measures. From a p-box $[F_l, F_u]$ for X one can infer bounds on the expectation for $f(X)$ by $\langle f \rangle_l = \inf_{F_l \le F \le F_u} \int_\Omega f dF$, $\langle f \rangle_u = \sup_{F_l \le F \le F_u} \int_\Omega f dF$, regarding $F_l \le F \le F_u$ as a set of measures, e.g., [47], [96]. The bounds can be computed numerically by discretization and formulation of a linear programming problem (LP), cf. [93].

5 Bayesian inference

As soon as incomplete information is based on subjective knowledge and can be updated iteratively by additional information, one can consider using Bayesian inference to handle uncertainties. Bayesian inference means reasoning on the basis of Bayes' rule, working with conditional probabilities.

Here we have the crucial problem, how to select a suitable prior distribution. For a reasonable choice real statistical data is needed in sufficient amount. Additionally, of course, incoming new observations are required for updating. Priors can be chosen, e.g., with the maximal entropy principle, cf. Section 2.2. In practice one often chooses a normal distribution to simplify calculations, or conjugate priors, i.e., a distribution where the posterior has a similar shape like the prior except from a change in some parameters. Actually, it is a well-known criticism that the

choice of the prior often seems to be quite arbitrary and merely in the will of the statistician.

A typically employed model in Bayesian inference is a so-called Bayesian or belief network (BN). A BN is a directed acyclic graph (DAG) between states of a system and observables. A node $\mathcal{N}$ and its parent nodes in the DAG represent the input information of the network which consists of tables of conditional probabilities of $\mathcal{N}$ conditional on its parent nodes. The whole DAG represents the joint distribution of all involved variables, even in higher dimensional situations. Computations using BNs can be done efficiently on the DAG structure, assumed that all conditional probabilities are precisely known.

What if the conditional PDFs of the tables of conditional probabilities in BNs are unknown or not precisely known? This happens frequently in practice, in particular for variables conditional on multiple further variables. The Bayesian approach appears to become useless in this case. A generalized approach to BNs with imprecise probabilities can be studied on the basis of so-called **credal networks**, e.g., [11], [35]. A credal network is a set of BNs with the same DAG structure, but imprecise values in the conditional probability tables. The probabilities can be given as intervals, or more generally described.

The Bayesian approach applies in design optimization, cf. [103]. Similar to RBDO (14) one minimizes a certain objective like design cost subject to probabilistic constraints involving the failure distribution. The associated joint distribution is estimated and updated from available data, starting with conjugate priors.

6 Dempster-Shafer theory

Dempster-Shafer theory enables to process incomplete uncertainty information allowing to compute bounds for failure probabilities and reliability.

We start with defining fuzzy measures, cf. [90]. A **fuzzy measure** $\widetilde{\mu} : 2^{\Omega} \to [0,1]$, fulfills

$$\widetilde{\mu}(\emptyset) = 0, \widetilde{\mu}(\Omega) = 1, \tag{15}$$

$$A \subseteq B \Rightarrow \widetilde{\mu}(A) \leq \widetilde{\mu}(B). \tag{16}$$

The main difference to a probability measure is the absence of additivity. Instead, fuzzy measures only satisfy monotonicity (16). To find lower and upper bounds for an unknown probability measure given incomplete information one seeks two fuzzy measures **belief** Bel and **plausibility** Pl, where Bel is a fuzzy measure with $\mathrm{Bel}(A \cup B) \geq \mathrm{Bel}(A) + \mathrm{Bel}(B) - \mathrm{Bel}(A \cap B)$, and Pl is a fuzzy measure with $\mathrm{Pl}(A \cup B) \leq \mathrm{Pl}(A) + \mathrm{Pl}(B) - \mathrm{Pl}(A \cap B)$.

To construct the measures Bel and Pl from the given uncertainty information one formalizes the information as a so-called **basic probability assignment** m : $2^{\Omega} \to [0,1]$ on a finite set $\mathcal{A} \subseteq 2^{\Omega}$ of non-empty subsets A of Ω, such that

$$m(A) \begin{cases} > 0 & \text{if } A \in \mathcal{A}, \\ = 0 & \text{otherwise,} \end{cases} \tag{17}$$

and the normalization condition $\sum_{A\in\mathcal{A}} m(A) = 1$. Sometimes m is also called **basic belief assignment**.

The basic probability assignment m is interpreted as the exact belief focussed on A, and not in any strict subset of A. The sets $A \in \mathcal{A}$ are called **focal sets**. The structure $(m, \mathcal{A})$, i.e., a basic probability assignment together with the related set of focal sets, is called a **Dempster-Shafer structure** (DS structure).

Given a DS structure $(m, \mathcal{A})$ we can construct Bel and Pl by

$$\mathrm{Bel}(B) = \sum_{\{A\in\mathcal{A}|A\subseteq B\}} m(A), \tag{18}$$

$$\mathrm{Pl}(B) = \sum_{\{A\in\mathcal{A}|A\cap B\neq\emptyset\}} m(A) \tag{19}$$

for $B \in 2^\Omega$.

Thus Bel and Pl have the sought property $\mathrm{Bel} \le \mathrm{Pr} \le \mathrm{Pl}$ by construction and, moreover, satisfy $\mathrm{Bel}(B) = 1 - \mathrm{Pl}(B^c)$. The information contained in the two measures Bel and Pl induced by the DS structure is often called a **random set**.

In the classical case the additivity of non-fuzzy measures would yield $\mathrm{Pl}(B) = 1 - \mathrm{Pl}(B^c) = \mathrm{Bel}$. Thus $\mathrm{Bel} = \mathrm{Pr} = \mathrm{Pl}$ and classical probability theory becomes a special case of DS theory. Also note that if we have a DS structure on the singletons of a finite Ω, then we have full stochastic knowledge equivalent to a CDF.

DS structures can be obtained from expert knowledge or in lower dimensions from histograms, or from the Chebyshev inequality $\Pr(|X-\mu| \le r) > 1 - \frac{\sigma^2}{r^2}$ given expectation value μ and variance σ^2 of a random variable X, cf. [75], [76], [77]: Let $r = \frac{\sigma}{\sqrt{1-\alpha}}$ for a fixed confidence level α, then $\Pr(\{|X-\mu| \le \frac{\sigma}{\sqrt{1-\alpha}}\}) > \alpha$. The sets $C_\alpha := \{\omega \in \Omega \mid |X(\omega) - \mu| \le \frac{\sigma}{\sqrt{1-\alpha}}\}$ for different values of α define focal sets, and we get Belief and Plausibility measures by $\mathrm{Bel}(C_\alpha) = \alpha$ and $\mathrm{Pl}(C_\alpha^c) = 1 - \alpha$, respectively.

To extend one-dimensional focal sets to the multi-dimensional case one can generate joint DS structures from the Cartesian product of marginal basic probability assignments assuming random set independence, cf. [10], or from weighting the 1-dimensional marginal focal sets, cf. [27]. In [103] we find the suggestion to employ Bayesian techniques to estimate and update DS structures from little amount of information.

To combine different, or even conflicting DS structures $(m_1, \mathcal{A}_1)$, $(m_2, \mathcal{A}_2)$ (in case of multiple bodies of evidence, e.g., several different expert opinions) to a new basic probability assignment m_{new} one uses Dempster's rule of combination [12], forming the basis of **Dempster-Shafer theory** or **evidence theory** [86],

$$m_{\mathrm{new}}(B) = \sum_{\{A_1\in\mathcal{A}_1, A_2\in\mathcal{A}_2|A_1\cap A_2=B\}} \frac{m_1(A_1)m_2(A_2)}{K} \tag{20}$$

with the normalization constant $K = 1 - \sum_{\{A_1\in\mathcal{A}_1, A_2\in\mathcal{A}_2|A_1\cap A_2=\emptyset\}} m_1(A_1)m_2(A_2)$ which is interpreted as the **conflict**.

The combination rule enables to compute a joint DS structure. Also note that the combination rule is a generalization of Bayes' rule, motivated by the criticism that a single probability assignment cannot model the amount of evidence one has.

The complexity of the rule is strongly increasing in higher dimensions, and in many cases requires independence assumptions for simplicity reasons avoiding problems with interacting variables. It is not yet understood how the dimensionality issue can be solved. Working towards more efficient computational implementations of evidence theory it can be attempted to decompose the high-dimensional case in lower dimensional components which leads to so-called compositional models, cf. [41].

The **extension** of a function f is based on the joint DS structure $(m, \mathcal{A})$. The new focal sets of the extension are $B_i = f(A_i)$, $A_i \in \mathcal{A}$, the new basic probability assignment is $m_{\text{new}}(B_i) = \sum_{\{A_i \in \mathcal{A} | f(A_i) = B_i\}} m(A_i)$.

To embed DS theory in design optimization one formulates a constraint on the upper bound of the failure probability p_f which should be smaller than an admissible failure probability p_a, i.e., $\text{Pl}(\mathbb{F}) \leq p_a$, for the failure set $\mathbb{F}$. This can be studied in [66] as evidence based design optimization (EBDO). One can also find further direct applications in engineering computing, e.g., in [29], [75].

DS structures enable to construct p-boxes [7], [15], [93], i.e., to determine lower bounds F_l and upper bounds F_u of the CDF of a random variable X,

$$F_l(t) = \text{Bel}(\{\omega \in \Omega \mid X(\omega) \leq t\}),$$
$$F_u(t) = \text{Pl}(\{\omega \in \Omega \mid X(\omega) \leq t\}).$$

Conversely it is possible to generate a DS structure that approximates a given p-box discretely, cf. [2], [13], [25]. Fix some levels $\alpha_1 \leq \alpha_2 \leq \cdots \leq \alpha_N = 1$ of the p-box, then generate focal sets by

$$A_i := [\inf\{x \mid F_u(x) = \alpha_i\}, \inf\{x \mid F_l(x) = \alpha_i\}], \tag{21}$$
$$m(A_1) = \alpha_1, m(A_i) = \alpha_i - \alpha_{i-1}, i = 2, \ldots, N.$$

Another relation to a different uncertainty representation concerns nested focal sets, i.e., $\mathcal{A} = \{A_1, A_2, \ldots, A_m\}$, $A_1 \subseteq A_2 \subseteq \cdots \subseteq A_m$. In this case

$$\text{Bel}(A \cap B) = \min(\text{Bel}(A), \text{Bel}(B)), \tag{22}$$
$$\text{Pl}(A \cup B) = \max(\text{Pl}(A), \text{Pl}(B)). \tag{23}$$

For nested focal sets the fuzzy measures Bel and Pl directly correspond to possibility and necessity measures, respectively, which appear in fuzzy set theory, cf. [16], as we will see in the next section.

We have learned that DS structures can unify several different uncertainty models, see, e.g., [48], but cannot overcome the curse of dimensionality being prohibitively expensive in higher dimensions.

7 Fuzzy sets

The development of fuzzy sets has started roughly in parallel to the development of DS theory with the goal to model vague verbal descriptions in absence of any statistical data. It is a generalization of conventional set theory redefining the characteristic function of a set A by a so-called **membership function** μ_A. The value $\mu_A(x)$ indicates the membership value of an uncertain variable x with respect to A. The value can be any real number between 0 and 1 as opposed to the characteristic function $1_A(x)$ which only takes binary values. A **fuzzy set** is a set A together with its related membership function μ_A.

This section will give a short overview on fuzzy sets, focussing on their application for uncertainty handling. The following terms play an important role in the theory of fuzzy sets. The **height** h of a fuzzy set is defined by $h := \max_x \mu_A(x)$. The **support** of a fuzzy set is the set $\{x \mid \mu_A(x) \neq 0\}$. The **core** or **modal values** of a fuzzy set is the set $\{x \mid \mu_A(x) = 1\}$. The **α-cut** C_α of a fuzzy set for a fixed value $\alpha \in [0,1]$ is the set

$$C_\alpha := \{x \mid \mu_A(x) \geq \alpha\}. \tag{24}$$

The α-cut is determined by the values of the membership function. Conversely one can construct μ_A from the knowledge of the α-cuts, cf. [102], to achieve an α-cut based representation of a fuzzy set:

$$\mu_A(x) = \sup_\alpha \min(\alpha, 1_{C_\alpha}(x)). \tag{25}$$

Note the relationship between BPA-structures on nested focal sets, cf. Section 6, and α-cuts of a fuzzy set with non-empty core, which are nested by definition, i.e., $C_\alpha \subseteq C_\beta$ for $\alpha \geq \beta$. Let $1 = \alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_N = 0$ be α-levels of a fuzzy set, then we can construct a BPA m on the α-cuts C_{α_i} by $m(C_{\alpha_i}) = \alpha_i - \alpha_{i+1}$, $i < N$, $m(C_{\alpha_N}) = \alpha_N$. Conversely a BPA-structure on nested focal sets $A_1 \subseteq A_2 \subseteq \cdots \subseteq A_N$ allows to construct a fuzzy set by $\alpha_N = m(A_N)$, $C_{\alpha_N} = A_N$, $\alpha_{N-1} = m(A_N) + m(A_{N-1})$, $C_{\alpha_{N-1}} = A_{N-1}$, $\ldots$, $\alpha_1 = \sum_{i=1}^N m(A_i) = 1$, $C_{\alpha_1} = A_1$, and then applying (25). Thus it is possible to convert expert knowledge modeled by a fuzzy set into a DS structure. Using the Dempster's rule, however, to combine different bodies of evidence in general leads to non-nested focal sets, hence a conversion back to the fuzzy set formalism is not possible after applying a combination rule.

Some special cases of fuzzy sets motivated the notation of fuzzy intervals and fuzzy numbers, cf. [101]. A **fuzzy interval** or **convex fuzzy set** is a fuzzy set with $\mu_A(x) \geq \min(\mu_A(a), \mu_A(b))$ for all $a, b, x \in [a, b]$. A **fuzzy number** is a fuzzy interval with closed α-cuts, compact support, and a unique modal value.

The definition of a fuzzy set and its membership function in higher dimensions is a straightforward generalization of the one-dimensional case. The extension of a function $f(x) = z$, $f : \mathbb{R}^n \to \mathbb{R}$, for a fuzzy set with membership function μ is constructed by the **extension principle** for a new membership function

$$\mu_{\text{new}}(z) = \sup_{x \in f^{-1}(z)} \mu(x), \tag{26}$$

cf. [101]. The construction involves an optimization problem with rapidly increasing complexity in higher dimensions.

It can be attempted to solve this problem by reduction of the problem to the α-cuts of the fuzzy set, cf. Section 7.1, or by sensitivity analysis, cf. Section 2.6.

Except from the dimensionality issue another criticism of fuzzy sets is the fact that the assignment of membership functions appears to be quite arbitrary, often defined by a single expert opinion. In lower dimensions membership functions can be estimated, e.g., from histograms, but there is no general, statistically well-grounded basis for the assignment of membership functions. Of course, if only vague verbal descriptions, i.e., highly informal uncertainty information, is available statistical properties are entirely absent. In this case, which represents the classical motivation of fuzzy sets, it can be argued that it is impossible to formulate a general recipe for processing the information. However, usually the information consists of a mixture of statistical and fuzzy descriptions, and conventional fuzzy methods cannot combine both. The concept of fuzzy randomness, cf. [54], [59], [63], [82], is one attempt of a combination.

The applications of fuzzy methods in engineering computing are vast. A famous application of fuzzy methods is fuzzy control, cf. [91]. Moreover, most design analyzing methods have their counterparts in the context of fuzzy sets, for instance, fuzzy reliability methods (e.g., [9], [63]), fuzzy differential equations (e.g., [28]), fuzzy finite element methods (e.g., [26], [60], [67]), fuzzy ARMA and other stochastic processes (e.g., [61]).

In fuzzy statistics, i.e., with sample points that are modeled as fuzzy numbers, one can apply statistical methods on non-precise data, cf. [94].

In design optimization fuzzy methods can be used to find clusters of permissible designs with fuzzy clustering methods, e.g., [38], [42]. Seeking the optimal design one can use fuzzy methods to compare different design points of different clusters with respect to some criterion, e.g., weighted distances from design constraints [3], [8], [40].

The following subsection presents a special fuzzy set based method which is highlighted because of its relationship to our approach based on clouds, cf. Section 9.

7.1 α-level optimization

The α-level optimization approach [62] is the most relevant fuzzy set based method for our purposes as it applies also in higher dimensional real-life situations and uses similar techniques as we will use employing the clouds formalism.

The α-level optimization method combines the α-cut representation (25) and the extension principle to determine the membership function μ_f of a function $f(\varepsilon)$, $f : \mathbb{R}^n \to \mathbb{R}$, given the membership function μ of the variable ε. This is achieved

by constructing the α-cuts $C_{f_{\alpha_i}}$ belonging to μ_f from the α-cuts C_{α_i} belonging to μ. To this end one solves the optimization problems

$$\min_{\varepsilon \in C_{\alpha_i}} f(\varepsilon), \tag{27}$$

$$\max_{\varepsilon \in C_{\alpha_i}} f(\varepsilon) \tag{28}$$

for different discrete values α_i. Finally from the solution f_{i_*} of (27) and f_i^* of (28) one constructs the α-cuts belonging to μ_f by $C_{f_{\alpha_i}} = [f_{i_*}, f_i^*]$.

To simplify the optimization step one assumes sufficiently nice behaving functions f and computationally nice fuzzy sets, i.e., convex fuzzy sets, typically triangular shaped fuzzy numbers.

In n dimensions one optimizes over a hypercuboid, obtained by the Cartesian product of the α-cuts $C_{\alpha_i} = C^1_{\alpha_i} \times C^2_{\alpha_i} \times \cdots \times C^n_{\alpha_i}$, where $C^j_{\alpha_i} := \{\varepsilon^j \mid \mu^j(\varepsilon^j) \geq \alpha_i\}$, $\mu^j(\varepsilon^j) := \sup_{\varepsilon^k, k \neq j} \mu(\varepsilon)$, $\varepsilon = (\varepsilon^1, \varepsilon^2, \ldots, \varepsilon^n)$. Here one has to assume non-interactivity of the uncertain variables $\varepsilon^1, \ldots, \varepsilon^n$.

Using a discretization of the α-levels by a finite choice of α_i the computational effort for this methods becomes tractable. From (25) one gets a step function for μ_f which is usually linearly approximated through the points f_{i_*} and f_i^* to generate a triangular fuzzy number.

8 Convex methods

Convex methods model uncertainty by so-called **anti-optimization** over convex sets, cf. [4], [19]. Assume that we wish to find the design point $\theta = (\theta^1, \theta^2, \ldots, \theta^{n_o})$ with the minimal design objective function value $g(\theta, \varepsilon)$, $g : \mathbb{R}^{n_o} \times \mathbb{R}^n \to \mathbb{R}$ under uncertainty of the vector of input variables ε. Also assume that the uncertainty of ε is described by a convex set C. Anti-optimization means finding the worst-case scenario for a fixed design point θ by the solution of an optimization problem of the type

$$\begin{aligned} &\max_{\varepsilon} \; g(\theta, \varepsilon) \\ &\text{s.t.} \;\; \varepsilon \in C \end{aligned} \tag{29}$$

The corresponding design optimization problem would be

$$\begin{aligned} &\min_{\theta} \; \max_{\varepsilon} \; g(\theta, \varepsilon) \\ &\text{s.t.} \qquad \varepsilon \in C \\ &\qquad\qquad \theta \in T \end{aligned} \tag{30}$$

where T is the set of possible selections for the design θ. As the inner level of problem (30), i.e., equation (29), maximizes the objective which is sought to be minimized for the design optimization in the outer level (i.e., one seeks the design

with minimal worst-case), the term *anti-optimization* has been proposed for this approach.

Investigating convex regions for the worst-case search is motivated by the fact that in many cases the level sets of probability densities are convex sets, e.g., ellipsoids for normal distributions. In this respect the term convex uncertainty for a random vector $\varepsilon \in \mathbb{R}^n$ is characterized by a convex set $C = \{\varepsilon \mid Q(\varepsilon) \leq c\}$, where Q is a quadratic form and ε is known to belong to C with some confidence. The quadratic form could be, e.g., $Q(\varepsilon) = (\varepsilon - m)^T C^{-1}(\varepsilon - m)$ with a vector of nominal values m and an estimated covariance matrix C.

Once one has a description by convex uncertainty one can apply optimization methods which can make convex methods applicable even in higher dimensions.

It should be remarked that this particular idea is one of the inspirations for the potential clouds concept, see the next section, where the potential function will be constructed to have convex level sets.

9 Potential clouds

This section gives an intuitive introduction of uncertainty representation by means of clouds [71], inspired by and combining ideas from p-boxes, random and fuzzy sets, convex methods, interval and optimization methods. We will learn in this section about the natural approach that has lead to use clouds for uncertainty modeling, handling incomplete information in higher dimensions, and to weave the methodology into an optimization problem formulation similar to (30).

The goal is to construct confidence regions in which we should be able to search for worst-case scenarios via optimization techniques. The construction should be possible on the basis of scarce, high-dimensional data, incomplete information, unformalized knowledge and information updates. As mentioned in previous sections, in lower dimensions and provided real empirical data one has powerful tools, like KS, e.g., to bound the CDF of a random variable X. What could one do to tackle the same problems for higher dimensional random vectors $\varepsilon \in \mathbb{R}^n$ with little or no information available? To generate data we will first simulate a data set and modify it with respect to the available uncertainty information. To reduce the dimensionality of the problem we will use a potential function $V : \mathbb{R}^n \to \mathbb{R}$. We will bound the CDF of $V(\varepsilon)$ using KS as in the one-dimensional case (like a p-box on $V(\varepsilon)$, cf. Section 4). From the bounds on the CDF of $V(\varepsilon)$ we get lower and upper confidence regions for $V(\varepsilon)$, and finally lower and upper confidence regions for ε as level sets of V.

Assume that we have a lower bound $\underline{\alpha}$ and an upper bound $\overline{\alpha}$ for the CDF F of $V(\varepsilon)$, $\underline{\alpha}$ continuous from the left and monotone, $\overline{\alpha}$ continuous from the right and monotone. Then we find nested lower and upper confidence regions for ε by: $\underline{C}_\alpha := \{x \in \mathbb{R}^n \mid V(x) \leq \underline{V}_\alpha\}$ if $\underline{V}_\alpha := \min\{V_\alpha \in \mathbb{R} \mid \overline{\alpha}(V_\alpha) = \alpha\}$ exists, and $\underline{C}_\alpha := \emptyset$ otherwise; analogously $\overline{C}_\alpha := \{x \in \mathbb{R}^n \mid V(x) \leq \overline{V}_\alpha\}$ if $\overline{V}_\alpha := \max\{V_\alpha \in \mathbb{R} \mid \underline{\alpha}(V_\alpha) = \alpha\}$ exists, and $\overline{C}_\alpha := \mathbb{R}^n$ otherwise.

The regions $\underline{C}_\alpha$ and $\overline{C}_\alpha$ are lower and upper confidence regions in the following

sense: the region $\underline{C}_\alpha$ contains at most a fraction of α of all possible values of ε in $\mathbb{R}^n$, since $\Pr(\varepsilon \in \underline{C}_\alpha) \leq \Pr(\overline{\alpha}(V(\varepsilon)) \leq \alpha) \leq \Pr(F(V(\varepsilon)) \leq \alpha) = \alpha$; analogously $\overline{C}_\alpha$ contains at least a fraction of α of all possible values of ε in $\mathbb{R}^n$. Generally holds $\underline{C}_\alpha \subseteq \overline{C}_\alpha$.

The interval-valued mapping $x \to [\underline{\alpha}(V(x)), \overline{\alpha}(V(x))]$ is called a **potential cloud**.

Note that potential clouds extend the p-box concept to the multivariate case without the exponential growth of work in the conventional p-box approach. From the fact that we construct a p-box on $V(\varepsilon)$ one can also see the relation to DS structures generated from p-boxes as in (21), with $A_i = \overline{C}_{\alpha_i} \backslash \underline{C}_{\alpha_i}$. Thus the focal sets are determined by the level sets of V. To see an interpretation in terms of fuzzy sets one may consider $\underline{C}_\alpha$, $\overline{C}_\alpha$ as α-cuts of a multi-dimensional interval valued membership function defined by $\underline{\alpha}$ and $\overline{\alpha}$. However, clouds allow for probabilistic statements, so they become a more powerful tool in the estimation of failure probabilities.

The potential clouds approach not only helps us to overcome the curse of dimensionality in real-life applications, but also it turns out to enable a flexible uncertainty representation. It can process incomplete knowledge of different kinds and allows for an adaptive interaction between the uncertainty elicitation and the optimization phase, reducing the incompleteness of epistemic information via information updating. The adaptive step is realized by a modification of the shape of V in a graphical user interface. This is a unique feature in higher dimensions.

To illustrate how unformalized knowledge can be provided by clouds assume that a set of data points is given, but no formal information about the probability distribution is available, in particular no correlation information. Frequently an expert still has some knowledge about the dependence between the uncertain variables involved, and he may be able to provide linear constraints as shown in Figure 3. The lower and upper confidence regions constructed with clouds then become polyhedra, cf. Figure 4. We also see that these regions reasonably approximate the confidence regions in case that we knew the correlations exactly.

The basic concept of embedding our approach in a design optimization problem is as follows. The designing expert provides an underlying system model – e.g., given as a black box model – and all currently available uncertainty information on the input variables of the model. The information is processed to generate a cloud that provides a nested collection of regions of relevant scenarios parameterized by a confidence level α. Thus we produce safety constraints for the optimization. The optimization minimizes a certain objective function (e.g., cost, mass) subject to the safety constraints to account for the robustness of the design, and subject to the functional constraints which are represented by the system model. The results of the optimization, i.e., the automatically found optimal design point and the worst-case analysis, are returned to the expert, who is given an interactive possibility to provide additional uncertainty information afterwards and rerun the procedure. For further details on the construction of potential clouds and cloud based design optimization the interested reader is referred to [32] and [33].

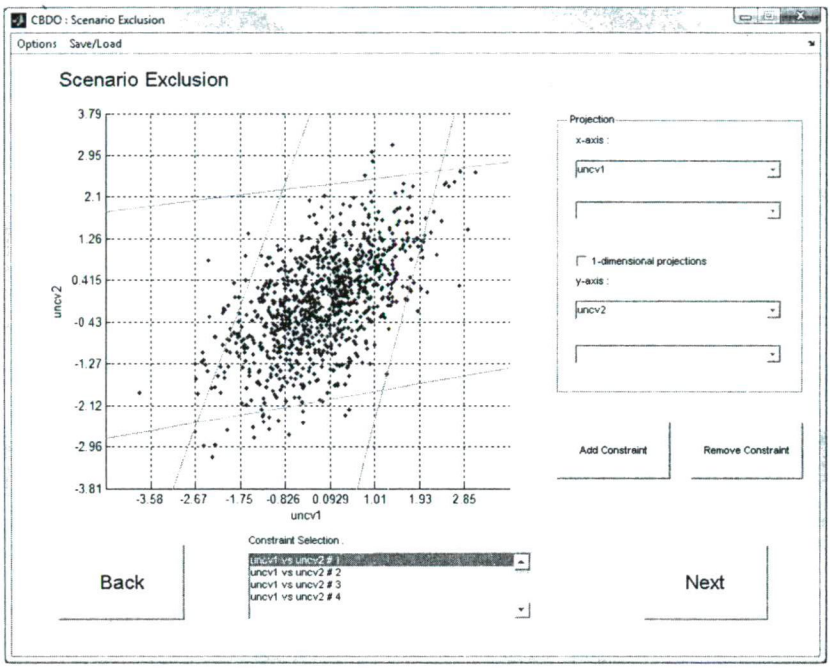


Figure 3: Two random variables ε^1, ε^2 with non zero correlation. The linear constraints model the unformalized knowledge of the expert about the dependence of the variables.

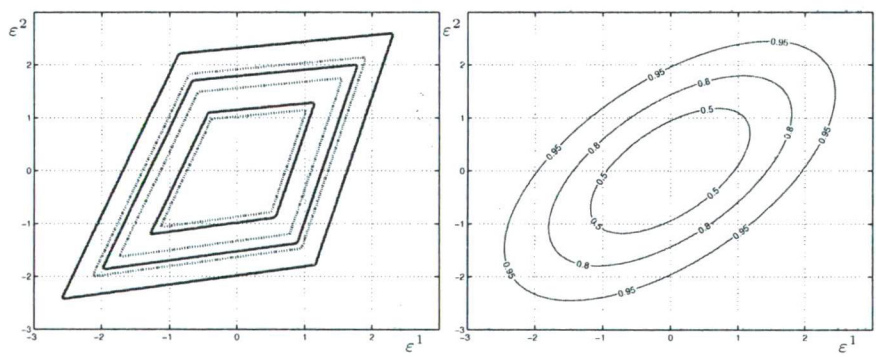


Figure 4: On the left, plotted with dotted and solid lines respectively, are the lower and upper confidence regions for $\alpha = 50\%, 80\%, 95\%$ of a 2-dimensional random variable $(\varepsilon^1, \varepsilon^2)$ belonging to a polyhedral potential cloud. On the right the corresponding confidence regions if the correlation was exactly known.

Acknowledgements

I would like to thank Arnold Neumaier who has significantly contributed to the creation of this paper with various comments. Also I would like to thank the anonymous reviewers for their fruitful remarks.

References

[1] Alexandrov, N.M. and Hussaini, M.Y. Multidisciplinary design optimization: State of the art. In *Proceedings of the ICASE/NASA Langley Workshop on Multidisciplinary Design Optimization*, Hampton, Virginia, USA, 1997.

[2] Aregui, A. and Denœux, T. Constructing predictive belief functions from continuous sample data using confidence bands. In *Proceedings of the 5th International Symposium on Imprecise Probability: Theories and Applications*, pages 11–19, Prague, Czech Republic, 2007.

[3] Beer, M., Liebscher, M., and Möller, B. Structural design under fuzzy randomness. In *Proceedings of the NSF Workshop on Reliable Engineering Computing*, pages 215–234, Savannah, Georgia, USA, 2004.

[4] Ben-Haim, Y. and Elishakoff, I. *Convex Models of Uncertainty in Applied Mechanics*. Elsevier, 1990.

[5] Berleant, D., Ferson, S., Kreinovich, V., and Lodwick, W.A. Combining interval and probabilistic uncertainty: Foundations, algorithms, challenges – an overview. In *Proceedings of the 4th International Symposium on Imprecise Probabilities and Their Applications*, Pittsburgh, Pennsylvania, USA, 2005.

[6] Berleant, D. and Xie, L. An interval-based tool for verified arithmetic on random variables of unknown dependency. Manuscript, 2005.

[7] Bernardini, A. *Whys and Hows in Uncertainty Modelling: Probability, Fuzziness and Anti-Optimization*, chapter What are random and fuzzy sets and how to use them for uncertainty modelling in engineering systems?, pages 63–125. Springer, 1999.

[8] Chen, S.H. Ranking fuzzy numbers with maximizing set and minimizing set. *Fuzzy Sets and Systems*, 17:113–129, 1985.

[9] Cheng, C.H. and Mon, D.L. Fuzzy system reliability analysis by interval of confidence. *Fuzzy Sets and Systems*, 56(1):29–35, 1993.

[10] Couso, I., Moral, S., and Walley, P. Examples of independence for imprecise probabilities. In *Proceedings of the 1st International Symposium on Imprecise Probabilities and Their Applications*, pages 121–130, Ghent, Belgium, 1999.

[11] Cozman, F.G. Credal networks. *Artificial Intelligence*, 120(2):199–233, 2000.

[12] Dempster, A.P. Upper and lower probabilities induced by a multivalued mapping. *Annals of Mathematical Statistics*, 38(2):325–339, 1967.

[13] Denœux, T. Constructing belief functions from sample data using multinomial confidence regions. *International Journal of Approximate Reasoning*, 42(3):228–252, 2006.

[14] Der Kiureghian, A. and Dakessian, T. Multiple design points in first and second-order reliability. *Structural Safety*, 20(1):37–49, 1998.

[15] Destercke, S., Dubois, D., and Chojnacki, E. Relating practical representations of imprecise probabilities. In *Proceedings of the 5th International Symposium on Imprecise Probability: Theories and Applications*, pages 155–163, Prague, Czech Republic, 2007.

[16] Dubois, D. and Prade, H. *Possibility Theory: An Approach to Computerized Processing of Uncertainty.* New York: Plenum Press, 1986.

[17] Eldred, M.S., Brown, S.L., Adams, B.M., Dunlavy, D.M., Gay, D.M., Swiler, L.P., Giunta, A.A., Hart, W.E., Watson, J.-P., Eddy, J.P., Griffin, J.D., Hough, P.D., Kolda, T.G., Martinez-Canales, M.L., and Williams, P.J. DAKOTA, A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis: Version 4.0 Users Manual. Sand Report SAND2006-6337, Sandia National Laboratories, 2006.

[18] Elishakoff, I. *Whys and Hows in Uncertainty Modelling: Probability, Fuzziness and Anti-Optimization*, chapter What may go wrong with probabilistic methods?, pages 265–283. Springer, 1999.

[19] Elishakoff, I. *Whys and Hows in Uncertainty Modelling: Probability, Fuzziness and Anti-Optimization*, chapter Are probabilistic and anti-optimization approaches compatible?, pages 263–355. Springer, 1999.

[20] Ferson, S. What Monte Carlo methods cannot do. *Human and Ecological Risk Assessment*, 2:990–1007, 1996.

[21] Ferson, S. *Ramas Risk Calc 4.0 Software: Risk Assessment with Uncertain Numbers.* Lewis Publishers, U.S., 2002.

[22] Ferson, S., Ginzburg, L., and Akcakaya, R. Whereof one cannot speak: When input distributions are unknown. *Risk Analysis*, 1996. In press, available online at: `http://www.ramas.com/whereof.pdf`.

[23] Ferson, S., Ginzburg, L., Kreinovich, V., Longpre, L., and Aviles, M. Exact bounds on finite populations of interval data. *Reliable Computing*, 11(3):207–233, 2005.

[24] Ferson, S., Ginzburg, L., Kreinovich, V., and Lopez, J. Absolute bounds on the mean of sum, product, etc.: A probabilistic extension of interval arithmetic. In *Extended Abstracts of the 2002 SIAM Workshop on Validated Computing*, pages 70–72, Toronto, Canada, 2002.

[25] Ferson, S., Kreinovich, V., Ginzburg, L., Myers, D.S., and Sentz, K. Constructing probability boxes and Dempster-Shafer structures. Sand Report SAND2002-4015, Sandia National Laboratories, 2003. Available on-line at `http://www.sandia.gov/epistemic/Reports/SAND2002-4015.pdf`.

[26] Fetz, T. Finite element method with fuzzy parameters. In *Proceedings of the IMACS Symposium on Mathematical Modelling*, volume 11, pages 81–86, Vienna, Austria, 1997.

[27] Fetz, T. Sets of joint probability measures generated by weighted marginal focal sets. In *Proceedings of the 2nd International Symposium on Imprecise Probabilities and Their Applications*, pages 171–178, Maastricht, The Netherlands, 2001.

[28] Fetz, T., Oberguggenberger, M., Jager, J., Koll, D., Krenn, G., Lessmann, H., and Stark, R.F. Fuzzy models in geotechnical engineering and construction management. *Computer-Aided Civil and Infrastructure Engineering*, 14(2):93–106, 1999.

[29] Fetz, T., Oberguggenberger, M., and Pittschmann, S. Applications of possibility and evidence theory in civil engineering. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 8(3):295–309, 2000.

[30] Filho, R.S., Cozman, F.G., Trevizan, F.W., de Campos, C.P., and de Barros, L.N. Multilinear and integer programming for Markov decision processes with imprecise probabilities. In *Proceedings of the 5th International Symposium on Imprecise Probability: Theories and Applications*, pages 395–403, Prague, Czech Republic, 2007.

[31] Fuchs, M., Girimonte, D., Izzo, D., and Neumaier, A. *Robust intelligent systems*, chapter Robust and automated space system design, pages 251–272. Springer, 2008.

[32] Fuchs, M. and Neumaier, A. Autonomous robust design optimization with potential clouds. *International Journal of Reliability and Safety, Special Issue on Reliable Engineering Computing*, 2008. Accepted, preprint available online at: `http://www.martin-fuchs.net/publications.php`.

[33] Fuchs, M. and Neumaier, A. Potential based clouds in robust design optimization. *Journal of Statistical Theory and Practice, Special Issue on Imprecision*, 2008. Accepted, preprint available on-line at: `http://www.martin-fuchs.net/publications.php`.

[34] Grandy, W.T. and Schick, L.H. *Maximum Entropy and Bayesian Methods.* Springer, 1990.

[35] Haenni, R. Climbing the hills of compiled credal networks. In *Proceedings of the 5th International Symposium on Imprecise Probability: Theories and Applications*, pages 213–221, Prague, Czech Republic, 2007.

[36] Hastings, W.K. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[37] Hohenbichler, M. and Rackwitz, R. Improvement of second-order reliability estimates by importance sampling. *Journal of Engineering Mechanics*, 114(12):2195–2199, 1988.

[38] Höppner, F., Klawonn, F., Kruse, R., and Runkler, T. *Fuzzy Cluster Analysis: Methods for Classification, Data Analysis and Image Recognition.* Wiley, 1999.

[39] Huyer, W. and Neumaier, A. SNOBFIT – Stable Noisy Optimization by Branch and Fit. *ACM Transactions on Mathematical Software*, 35(2), 2008. Article 9, 25 pages.

[40] Jain, R. Decision making in the presence of fuzzy variables. *IEEE Transactions on Systems, Man, and Cybernetics*, 6(10):698–703, 1976.

[41] Jirousek, R., Vejnarova, J., and Daniel, M. Compositional models of belief functions. In *Proceedings of the 5th International Symposium on Imprecise Probability: Theories and Applications*, pages 243–251, Prague, Czech Republic, 2007.

[42] Kaufman, L. and Rousseeuw, P.J. *Finding Groups in Data: An Introduction to Cluster Analysis.* Wiley, 1990.

[43] Kaymaz, I. and Marti, K. Reliability-based design optimization for elastoplastic mechanical structures. *Computers & Structures*, 85(10):615–625, 2007.

[44] Kijawatworawet, W., Pradlwarter, H.J., and Schueller, G.I. Structural reliability estimation by adaptive importance directional sampling. In *Structural Safety and Reliability: Proceedings of ICOSSAR'97*, pages 891–897, Balkema, 1998.

[45] Koch, P.N., Simpson, T.W., Allen, J.K., and Mistree, F. Statistical approximations for multidisciplinary optimization: The problem of size. *Special Issue on Multidisciplinary Design Optimization of Journal of Aircraft*, 36(1):275–286, 1999.

[46] Kolmogorov, A. Confidence limits for an unknown distribution function. *The Annals of Mathematical Statistics*, 12(4):461–463, 1941.

[47] Kozine, I. and Krymsky, V. Enhancement of natural extension. In *Proceedings of the 5th International Symposium on Imprecise Probability: Theories and Applications*, pages 253–261, Prague, Czech Republic, 2007.

[48] Kreinovich, V. *Random Sets: Theory and Applications*, chapter Random sets unify, explain, and aid known uncertainty methods in expert systems, pages 321–345. Springer, 1997.

[49] Kreinovich, V. Probabilities, intervals, what next? optimization problems related to extension of interval computations to situations with partial information about probabilities. *Journal of Global Optimization*, 29(3):265–280, 2004.

[50] Kreinovich, V., Beck, J., Ferregut, C., Sanchez, A., Keller, G.R., Averill, M., and Starks, S.A. Monte-Carlo-type techniques for processing interval uncertainty, and their engineering applications. In *Proceedings of the NSF Workshop on Reliable Engineering Computing*, pages 139–160, Savannah, Georgia, USA, 2004.

[51] Kreinovich, V. and Ferson, S. A new Cauchy-based black-box technique for uncertainty in risk analysis. *Reliability Engineering & System Safety*, 85(1–3):267–279, 2004.

[52] Kreinovich, V., Ferson, S., and Ginzburg, L. Exact upper bound on the mean of the product of many random variables with known expectations. *Reliable Computing*, 9(6):441–463, 2003.

[53] Kreinovich, V. and Trejo, R. *Handbook of Randomized Computing*, chapter Error estimations for indirect measurements: randomized vs. deterministic algorithms for 'black-box' programs, pages 673–729. Kluwer, 2001.

[54] Kwakernaak, H. Fuzzy random variables – ii: Algorithms and examples for the discrete case. *Information Sciences*, 17:253–278, 1979.

[55] Makino, K. and Berz, M. Efficient control of the dependency problem based on taylor model methods. *Reliable Computing*, 5(1):3–12, 1999.

[56] Marti, K. and Kaymaz, I. Reliability analysis for elastoplastic mechanical structures under stochastic uncertainty. *Zeitschrift fr Angewandte Mathematik und Mechanik*, 86(5):358–384, 2006.

[57] McKay, M.D., Conover, W.J., and Beckman, R.J. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 221:239–245, 1979.

[58] Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., and Teller, E. Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, 21(6):1087–1092, 1953.

[59] Möller, B. and Beer, M. *Fuzzy Randomness: Uncertainty in Civil Engineering and Computational Mechanics.* Springer-Verlag Berlin Heidelberg, 2004.

[60] Möller, B., Beer, M., Graf, W., and Sickert, J. U. Fuzzy finite element method and its application. In *Trends in Computational Structural Mechanics*, pages 529–538, Barcelona, Spain, 2001.

[61] Möller, B., Beer, M., and Reuter, U. Theoretical basics of fuzzy-randomness – application to time series with fuzzy data. In *Safety and Reliability of Engineering Systems and Structures: Proceedings of the 9th International Conference on Structural Safety and Reliability*, Rome, Italy, 2005.

[62] Möller, B., Graf, W., and Beer, M. Fuzzy structural analysis using α-level optimization. *Computational Mechanics*, 26(6):547–565, 2000.

[63] Möller, B., Graf, W., and Beer, M. Fuzzy probabilistic method and its application for the safety assessment of structures. In *Proceedings of the European Conference on Computational Mechanics*, Cracow, Poland, 2001.

[64] Moore, R.E. *Methods and Applications of Interval Analysis.* Society for Industrial & Applied Mathematics, 1979.

[65] Mourelatos, Z.P. and Liang, J. An efficient unified approach for reliability and robustness in engineering design. In *Proceedings of the NSF Workshop on Reliable Engineering Computing*, pages 119–138, Savannah, Georgia, USA, 2004.

[66] Mourelatos, Z.P. and Zhou, J. A design optimization method using evidence theory. *Journal of Mechanical Design*, 128(4):901–908, 2006.

[67] Muhanna, R.L. and Mullen, R.L. Formulation of fuzzy finite element methods for mechanics problems. *Computer-Aided Civil and Infrastructure Engineering*, 14(2):107–117, 1999.

[68] Muhanna, R.L. and Mullen, R.L. Uncertainty in mechanics problems – interval-based approach. *Journal of Engineering Mechanics*, 127(6):557–566, 2001.

[69] Nataf, A. Determination des distributions de probabilites dont les marges sont donnees. *Comptes Rendus de lAcademie des Sciences*, 225:42–43, 1962.

[70] Neumaier, A. *Interval Methods for Systems of Equations.* Cambridge University Press, 1990.

[71] Neumaier, A. Clouds, fuzzy sets and probability intervals. *Reliable Computing*, 10(4):249–272, 2004. Available on-line at:
`http://www.mat.univie.ac.at/~neum/ms/cloud.pdf`.

[72] Neumaier, A. Uncertainty modeling for robust verifiable design. Slides, 2004. Available on-line at: `http://www.mat.univie.ac.at/~neum/ms/uncslides.pdf`.

[73] Neumaier, A., Fuchs, M., Dolejsi, E., Csendes, T., Dombi, J., Banhelyi, B., and Gera, Z. Application of clouds for modeling uncertainties in robust space system design. ACT Ariadna Research ACT-RPT-05-5201, European Space Agency, 2007.

[74] Nguyen, H. Fuzzy sets and probability. *Fuzzy Sets and Systems, Social Issue on Fuzzy Sets: Where do we stand? Where do we go?*, 90(2):129–132, 1997.

[75] Oberguggenberger, M. and Fellin, W. Assessing the sensitivity of failure probabilities: a random set approach. In *Safety and Reliability of Engineering Systems and Structures: Proceedings of the 9th International Conference on Structural Safety and Reliability*, pages 1755–1760, Rome, Italy, 2005.

[76] Oberguggenberger, M. and Fellin, W. Reliability bounds through random sets: Non-parametric methods and geotechnical applications. *Computers and Structures*, 86(10):1093–1101, 2008.

[77] Oberguggenberger, M., King, J., and Schmelzer, B. Imprecise probability methods for sensitivity analysis in engineering. In *Proceedings of the 5th International Symposium on Imprecise Probability: Theories and Applications*, pages 317–325, Prague, Czech Republic, 2007.

[78] Papadrakakis, M. and Lagaros, N.D. Reliability-based structural optimization using neural networks and Monte Carlo simulation. *Computer Methods in Applied Mechanics and Engineering*, 191(32):3491–3507, 2002.

[79] Pownuk, A. Calculation of displacement in elastic and elastic-plastic structures with interval parameters. In *Proceedings of the 33rd Solid Mechanics Conference*, pages 5–9, Zakopane, Poland, 2000.

[80] Pownuk, A. General interval FEM program based on sensitivity analysis method. In *Proceedings of the 3rd International Workshop on Reliable Engineering Computing*, pages 397–428, Savannah, Georgia, USA, 2008.

[81] Press, W.H., Teukolsky, S.A., Vetterling, W.T., and Flannery, B.P. *Numerical Recipes in C.* Cambridge University Press, 2nd edition, 1992.

[82] Puri, M. and Ralescu, D. *Madan Lal Puri Selected Collected Works, Volume 3: Time Series, Fuzzy Analysis and Miscellaneous Topics*, chapter Fuzzy Random Variables. Brill Academic Pub, 2003.

[83] Rackwitz, R. Reliability analysis – a review and some perspectives. *Structural Safety*, 23(4):365–395, 2001.

[84] Rosenblatt, M. Remarks on a multivariate transformation. *Annals of Mathematical Statistics*, 23(3):470–472, 1952.

[85] Saad, E. Structural optimization based on evolution strategy. In *Advanced Computational Methods in Structural Mechanics*, pages 266–280, Barcelona, Spain, 1996.

[86] Shafer, G. *A Mathematical Theory of Evidence.* Princeton University Press, 1976.

[87] Shannon, C.E. and Weaver, W. *The Mathematical Theory of Communication.* University of Illinois Press, 1949.

[88] Skulj, D. *Soft Methods for Integrated Uncertainty Modelling*, chapter Finite Discrete Time Markov Chains with Interval Probabilities, pages 299–306. Springer, 2006.

[89] Skulj, D. Regular finite Markov chains with interval probabilities. In *Proceedings of the 5th International Symposium on Imprecise Probability: Theories and Applications*, pages 405–413, Prague, Czech Republic, 2007.

[90] Sugeno, M. *Theory of fuzzy integrals and its applications.* PhD thesis, Tokyo Institute of Technology, 1974.

[91] Sugeno, M. An introductory survey of fuzzy control. *Information Science*, 36:59–83, 1985.

[92] ter Marten, E.J.W., Doorn, T.S., Croon, J.A., Bargagli, A., di Bucchianico, A., and Wittich, O. Importance sampling for high speed statistical Monte-Carlo simulations. NXP Technical Note NXP-TN-2007-00238, NXP Semiconductors, 2007.

[93] Utkin, L. and Destercke, S. Computing expectations with p-boxes: two views of the same problem. In *Proceedings of the 5th International Symposium on Imprecise Probability: Theories and Applications*, pages 435–443, Prague, Czech Republic, 2007.

[94] Viertl, R. *Statistical Methods for Non-Precise Data.* CRC Press, 1996.

[95] Walley, P. *Statistical Reasoning with Imprecise Probability.* Chapman and Hall, 1991.

[96] Walley, P. Measures of uncertainty in expert systems. *Artificial Intelligence*, 83(1):1–58, 1996.

[97] Walley, P. Towards a unified theory of imprecise probability. *International Journal of Approximate Reasoning*, 24(2–3):125–148, 2000.

[98] Weisstein, E.W. Cauchy distribution. MathWorld – A Wolfram Web Resource, 2008. Available on-line at: `http://mathworld.wolfram.com/CauchyDistribution.html`.

[99] Williamson, R.C. *Probabilistic Arithmetic.* PhD thesis, University of Queensland, 1989.

[100] Williamson, R.C. and Downs, T. Probabilistic arithmetic. I. numerical methods for calculating convolutions and dependency bounds. *International Journal of Approximate Reasoning*, 4(2):89–158, 1990.

[101] Zadeh, L.A. Fuzzy sets. *Information and Control*, 8(3):338–353, 1965.

[102] Zadeh, L.A. Similarity relations and fuzzy orderings. *Information Sciences*, 3(2):177–200, 1971.

[103] Zhou, J. and Mourelatos, Z.P. Design under uncertainty using a combination of evidence theory and a Bayesian approach. In *Proceedings of the 3rd International Workshop on Reliable Engineering Computing*, pages 171–198, Savannah, Georgia, USA, 2008.

Acta Cybernetica 19 (2009) 93–103.

Cohesion and Balance in a Human Resource Allocation Problem

László Illyés*

Abstract

Collaborative work appears between intelligent agents of different types. The problem discussed occurred when many construction workers were taken to Germany from Romania to work in construction projects. Managers have to make independent groups of workers from some categories, like carpenters, brick layers, etc. To discover their collaborative attitudes they use the scoring method, where every worker scores the others from different trades. The objectives are to form groups of workers with high compatibility value and to have a high compatibility value for the worst group, too. The problem becomes more interesting if software collaborative groups or specialized intelligent agents are involved. One has to prospect also the level of knowledge overlap between the trade groups of agents. This paper resumes to the problem of construction workers so as there is no overlap between the trades and the level of knowledge is not in the universe of discourse. We propose a Greedy and a genetic algorithm approach and we compare these methods.

Keywords: human resource allocation, genetic algorithm, minimax problem

1 Introduction

Our world proceeds toward team work. Nobody can perform good tasks alone on a long term. Nitchi in [1] considers the collaboration as an intelligent activity based on 3C (Communication, Coordination and Cooperation). Collaborative work in a group needs compatibility analysis. One can perform this with personality and aptitude tests, but also with the scoring method, where the group actors score each other. This paper proposes to take in consideration a real-life problem that came up when many construction workers were taken to Germany from Romania to work in construction projects [2]. A managerial problem was to distribute them in independently working groups. And the objectives were that those groups should be balanced and with high compatibility factors. The meaning of 'balanced' in this case is that the least fitted group should have a maximum value.

*Sapientia Hungarian University of Transylvania from Cluj-Napoca, Faculty of Business and Humanities, Department of Mathematics and Informatics, Miercurea Ciuc.
E-mail: illyeslaszlo@sapientia.siculorum.ro

In this paper we take in consideration the next relaxations without a loss of generality: we use only the scoring method (the aptitudes are not tested), there is no overlap between trade workers (a carpenter does not perform bricklayer tasks), inside a trade there is no scoring.
We organize the remainder of the paper as follows. In section 2 we provide the mathematical background. Section 3 and 4 show the Greedy and the Genetic Algorithm approach. Experimental results are shown in section 5. Finally, we draw some conclusions to this paper.

2 The mathematical model

We consider N the total number of workers, n the cardinality of trades, m_i cardinality of workers in trade i, m_{min} minimal cardinality of m_i, c_{ij} the score given by person i to person j. We have $\sum_{i=1}^{n} m_i = N$, $c_{ij} = 1, \ldots, 5$. In the case when worker i is in a same trade with worker j we have $c_{ij} = 0$. Considering the scoring matrix c_{ij} we construct the symmetrical collaboration matrix $v_{ij} = v_{ji} = c_{ij} * c_{ji}$. We denote with W the total social value of the groups formed. The mathematical model obtained is:

$$\max\left(\sum_{j=1}^{N}\sum_{i=1}^{N} v_{ij}x_{ij}\right) \tag{1}$$

where $x_{ij} = 1$ in case of worker i is in a same group with worker j and $x_{ij} = 0$ otherwise.
The second objective function reflects the balance between the groups. This objective leads us to a max-min or min-max problem. In case we have the formed groups U_k with $k = 1, \ldots, m_{min}$, the second objective has the next form:

$$\max\left(\min_{k}\left\{\sum_{i,j\in U_k} v_{ij}\right\}\right) \tag{2}$$

For our testing purpose we choose an objective function that reflects both initial objectives. If we use the previous notations, the new mathematical model obtained will be:

$$\max\left(\left(\sum_{j=1}^{N}\sum_{i=1}^{N} v_{ij}x_{ij}\right) + \min_{k}\left(\sum_{j,j\in U_k} v_{ij}\right)\right) \tag{3}$$

The first part of the formula reflects the first objective, the second, the second objective.
The objective function can be extended into this more general form, where $\propto (m_{min})$ is a function of the minimal cardinality trade:

$$\max\left(\left(\sum_{j=1}^{N}\sum_{i=1}^{N} v_{ij}x_{ij}\right) + \propto (m_{min}) * \min_{k}\left(\sum_{j,j\in U_k} v_{ij}\right)\right) \tag{4}$$

We cannot reduce the whole problem to a minimax problem for many reasons. One of this appears when we can form more groups than needed.

3 The Greedy approach

Greedy algorithms, as the word suggests, mimic the behavior of the people, for example, in every moment they try to obtain the best result that the present situation offers. In this section we propose some Greedy approaches to solve the main problem or subproblems. We think that the Greedy coalition formation proposed by us is a very interesting one because it follows both objectives of the managers at the same time.

3.1 Greedy exclusion of workers

This procedure solves only a subproblem of the whole because after the exclusion of workers we have to try to proceed to the coalition formation. In case the workers cardinality in trades is not the same, one has to exclude some of the workers from the coalition formation. This is true even in cases when managers can form more groups than needed. The Greedy exclusion of workers can be applied to the score matrix c_{ij} or to the collaboration matrix v_{ij}. The Greedy exclusion using the score matrix means that we exclude the workers with the smallest sum of scores received from the others and who belong to trades with superior cardinality than m_{min}. The Greedy exclusion using the collaboration matrix means that we exclude from the same trades the workers with the smallest sum of collaborative values. We study the problem for a small numerical example having brute-force result to compare with the algorithm's results. Like we expected and in concordance with the defined objective functions, we find that the exclusion using the collaboration matrix performs better. We observe that both approaches can exclude the global optima.

3.2 Greedy coalition formation

This algorithm is considered to be in the focus of the paper. We consider the 'best individual' any worker who, included in the group, brings the highest collaboration value to that group. Algorithm 1 describes this approach. The Greedy approach for the first objective derives from Step 6 of the algorithm. The second objective is hidden in Steps 5 and 6. The worst group chooses first to gain some equilibrium.

When we construct the collaboration matrix we put the members of each trade in an adjacent position. This arrangement is useful for the implementation and understanding of the algorithm. As we see on the table presented, trades are {1,2,3,4}; {5,6,7,8} and {9,10,11}. Because of the last trade, we can form only 3 groups. In the first step, the algorithm gives the following three sub-groups: {9, 5}=20; {10, 2}=20 and {11, 6}=16. In the following step, the group with the

Algorithm 1 A Greedy-type algorithm

Funct GRW1

1: Choose one trade from the smallest cardinality trades.
2: Put each individual from that trade in a different group.
3: Initialize the group's collaboration values with 0.
4: **while** groups are not formed **do**
5: Put the coalitions in the ascending order of collaboration points gathered.
6: In the order obtained in the previous step, choose the 'best individual' from the remaining individuals belonging to the remaining groups.
7: Add the collaboration values between the new member and the old members to the collaboration value of the group.
8: **end while**

Table 1: One representation of a collaboration matrix

	1	2	3	4	5	6	7	8	9	10	11
1	0	0	0	0	2	12	10	6	4	5	3
2	0	0	0	0	6	3	5	5	2	20	1
3	0	0	0	0	15	25	1	3	2	6	3
4	0	0	0	0	5	2	8	15	4	12	6
5	2	6	15	*5*	0	0	0	0	20	3	3
6	12	3	*25*	2	0	0	0	0	15	5	16
7	10	*5*	1	8	0	0	0	0	12	12	5
8	6	5	3	15	0	0	0	0	12	5	8
9	4	2	2	*4*	*20*	15	12	12	0	0	0
10	5	*20*	6	12	3	5	*12*	5	0	0	0
11	3	1	*3*	6	3	*16*	5	8	0	0	0

smallest value will choose first. In the case of tie results, the algorithm chooses in a lexicographical order. The next step's results are the following groups: {11, 6, 3}=44; {9, 5, 4}=29 and {10, 2, 7}=37. In this case W=110 and the value of the objective function, named fitness function (that we will use in GA part) $F_f = W + V_{min} = 139$. The values used are emphasized in the matrix.

3.3 Greedy coalition formation when supply dominates

In this case we can form more groups than needed. The algorithm uses the previous algorithm, denoted with GRW1:

Algorithm 2 A Greedy-type algorithm with domination of supply

Funct GRWD

1: Solve the problem with GRW1.
2: Keep the best result
3: **while** cardinality of groups is greater than necessary **do**
4: Eliminate the worker from the smallest cardinality trade used at step 1 in GRW1 that is from the worst group.
5: Solve the reduced problem with GRW1.
6: Keep the best result so far.
7: **end while**

4 Genetic algorithm approach

A genetic algorithm (GA) is a heuristic method that mimics the evolution of species, described in [3] by Charles Darwin. Holland, in [4] describes the canonical genetic algorithm. In such an algorithm we use a population of individuals. Those are coded in binary strings (chromosomes) that describe the solution space of the problem. The first population is generated in a pseudo-randomized mode. This population is evolved by the GA in certain cycles, named generations. An accuracy value, named value of the fitness function gives us how 'good' the solution is. One can calculate the fitness value after decoding the chromosome. The best solution of the problem has the best fitness value too. This value is used in the reproduction process. The individuals with high fitness value have more chance to reproduce their genes. Standard implementation of the principle is the roulette wheel, fitness based selection. After the selection, the selected individuals are submitted to other genetic operators: crossover and mutation. The first operator combines two strings of chromosomes, the second modifies a single chromosome in a few bits. Important factors of efficiency of the GA are the probability values of applying these operators; the first contributes to the exploitation, the second to the exploration of the search space. De Jong in [5, 6, 7], gives some numbers for the 'optimal' parameters of the GA.

4.1 Chromosomes formed by independent gene chains

Permutation design of an individual (chromosome) is introduced to handle the TSP (Traveling Salesperson Problem). In this case we have special codification of chromosomes, every gene representing a specific city. The order of the cities in each chromosome represents the order of traveling, consequently genes are not substitutable, every gene has to be in all chromosomes of all generations. Because of this, we have specific genetic operators to maintain the structure of the chromosomes. We have crossover operators: PMX (Partially Mixed Crossover), OX (Order Crossover), CX (Cycle Crossover), ERC (Edge Recombination Crossover), Greedy Crossover and mutation operators, like the inversion operator or the classical operator of permutation mutation where two bits of the chromosome are switched.

The genetic representation of our problem space leads us to chains of permutation chromosomes that form the main chromosome. Every chain codes a trade of workers. Workers from the first trade are denoted with a_i. One chromosome could be the following:
$\{a_1, a_2, a_3, a_4, a_5; b_1, b_2, b_3, b_4, b_5; c_1, c_2, c_3, c_4\}$
The decoding of this chromosome means that the groups formed are $\{a_1, b_1, c_1\}$, $\{a_2, b_2, c_2\}$, $\{a_3, b_3, c_3\}$, $\{a_4, b_4, c_4\}$and a_5, b_5 are not used in the coalition formation. To roam the total search space we don't need to permute all the chains from the chromosome. A crossover of a chromosome will be in the following way: we fix one chain from the minimal cardinality trades (in our case c_i), the rest of the chains will be submitted to a classical permutation crossover operator separately. In case we use the PMX (Partially Mixed Crossover) with hot points in the 3 and 2 positions in the 1st and 2nd sub-chains with the next parents:
$P1 = \{a_1, a_2, a_3, \|, a_4, a_5; b_1, b_2, \|, b_3, b_4, b_5; c_1, c_2, c_3, c_4\}$
$P2 = \{a_5, a_2, a_1, \|, a_4, a_3; b_4, b_1, \|, b_3, b_5, b_2; c_1, c_2, c_3, c_4\}$ the offsprings are:
$O1 = \{a_1, a_2, a_3, \|, a_5, a_4; b_1, b_2, \|, b_4, b_5, b_3; c_1, c_2, c_3, c_4\}$ and
$O2 = \{a_5, a_2, a_1, \|, a_3, a_4; b_4, b_1, \|, b_2, b_3, b_5; c_1, c_2, c_3, c_4\}$
If we decode the 2 offsprings they give us the next coalitions:
$\{a_1, b_1, c_1\}$, $\{a_2, b_2, c_2\}$, $\{a_3, b_4, c_3\}$, $\{a_5, b_5, c_4\}$, a_4 and b_3 are excluded
$\{a_5, b_4, c_1\}$, $\{a_2, b_1, c_2\}$, $\{a_1, b_2, c_3\}$, $\{a_3, b_3, c_4\}$, a_4 and b_5 are excluded
We use the mutation operator over every sub-chain. The probability of mutation for the whole chromosome is between 0.1% and 1%. We assume that the algorithm choses for mutation from the first trade, workers from positions 2 and 4:
$O2 = \{a_5, a_2, a_1, a_3, a_4; b_4, b_1, b_2, b_3, b_5; c_1, c_2, c_3, c_4\}$
$O2' = \{a_5, a_3, a_1, a_2, a_4; b_4, b_1, b_2, b_3, b_5; c_1, c_2, c_3, c_4\}$

4.2 Chromosome representation with control genes

Control genes mimic real gene structure behavior [8]. Every chromosome is formed by a chain of binary control genes and the permutation chains, representing the trades. Control genes can activate or deactivate the functionality of the corresponding permutation chain. In this problem, if the control gene for a chain is '1', then crossover and mutation genetic operators are performed on that chain (or trade) and no operator is applied in the other case. The complex crossover operator has two parts, one for control genes and one for the permutation chain. Since every chromosome has a control gene structure, when we apply the crossover operator we randomly choose only one of the control genes from the two parents. In case one control gene is 'not connected' (0), the first offspring inherits the whole corresponding sub-chain from the first parent without modification, and the second offspring inherits in the same way from the second parent. The 'connected' (where control gene is 1) sub-chains are subject to permutation crossover operators. Those operators are applied to the same chains (trades) from the two parental chromosomes. One can use all binary genetic operators known to obtain the offspring's control genes. Because we don't want to have any more mutational effects for this combinatorial space, we use the one point crossover operator and a single-gene mutation

operator with small probability for the control genes.

Figure 1 shows an example of control gene structure. Trades formed by workers are $\{1,2,3,4\}$; $\{5,6,7,8,9\}$; $\{10,11,12,13\}$; $\{14,15,16,17\}$. Trades 2 and 3 are 'connected' and subject to genetic operators.

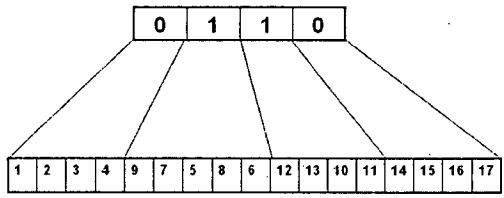


Figure 1: Example of control genes structure

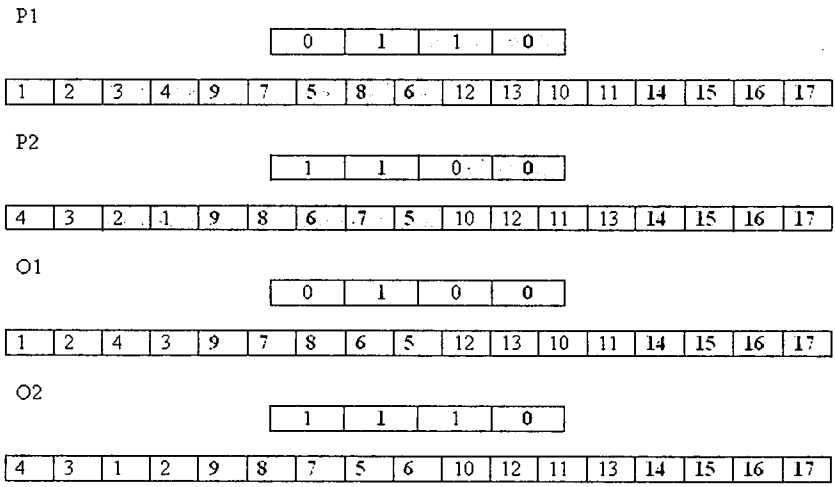


Figure 2: Complex crossover-PMX and single hot point

Figure 2 shows us how the complex crossover operator works: we have PMX operator for permutation chains when we use the second parent's control genes and single point crossover for the control chain. In this case the first and the second trades are subject to the crossover operator. We assume that the random generator chose the hot points between the 2nd and 3rd worker for both trades. The control genes are subject of a single point crossover operator with a hot point between the 2nd and 3rd genes. Trades are: $\{1,2,3,4\}$, $\{5,6,7,8,9\}$, $\{10,11,12,13\}$ and $\{14,15,16,17\}$. The fitness function used for testing is defined in the Mathematical model section.

5 Comparative results of the methods

5.1 Implementation

The program is implemented in Java language. The genetic algorithm part is implemented using the Jgap3.3[9], freely available GA package. The whole structure of the Jgap and a reused code from A. Mescauscas, N. Rotstan, K. Meffert is used. The main reused code is from TSP problem solved by A. Mescauscas[10]. We implement the cycle crossover operator and the composed crossover operator for the control gene structure reusing the Greedy crossover operator code. We also implement the fitness function for this problem. We use a population with 512 chromosomes, a generation number of 128, crossover rate of 100% and mutation rate of 1%.

5.2 Results obtained from the three data sets

We save the two generated data sets in files and we use those in the testing process. We have 20 workers and 4 trades in the first set and 120 workers and 6 trades in the second set. The third data set is the student database collection, where students were asked about their preferences and the trades are formed in a random process. The students had known each other for 2 years. An introduction to this problem is treated in [11].
GA with independent gene chains gives a better result only in the case of the student database (18 students) and maybe because the distribution is far from uniform one. Table 2 shows the results obtained for the student database. The Greedy solution of 492, at the bottom of the table is singular because the algorithm always chooses the best worker in lexicographical order. The genetic algorithms run 40 times to obtain these results and give us the min, max and avg values presented in the two columns inside the table for each type of GA (with or without control genes). All

Table 2: The students example results

	Genetic algorithm	Control genes GA
min	477.00	484.00
max	500.00	525.00
avg	488.90	502.90
Greedy	492.00	492.00

the results obtained with control gene structure are better than those obtained with the simple GA (with independent gene chains), and the simple GA performs better than the Greedy solution.
In the other cases of 20 and 120 workers the Greedy algorithm performs better. In order to increase the results of these cases, we combine the two methods. We generate the first population in the neighborhood of the Greedy solution. In this

case 'neighborhood' means that the first population chromosomes are derived from the Greedy solution chromosome by switching one pair of workers from a trade. If we want to define a greater neighborhood we can switch two or more pairs of workers.
The results obtained for the data set with 20 workers in 4 trades using the Greedy method, the simple GA (with independent gene chains), the GA with control genes and the combined Greedy+GA method are shown in Figure 3. The same results are shown for the 120 workers data with workers in 6 trades in Figure 4. As we see in both figures, the results obtained in the following ascendant order are: simple GA, GA with control genes, Greedy result and the mixed algorithm, when the first population of the GA algorithm is generated in the neighborhood of the Greedy solution. Like in the students case the Greedy method give here one result for each data set too. We note that if the workers cardinality increases, than the difference between the GA solutions with or without control genes are insignificant. The average values of both data sets results are better when we use the control-gene structure instead of independent gene chains. We can affirm that the algorithm with control-gene structure performs better than the simple GA with independent gene chains.

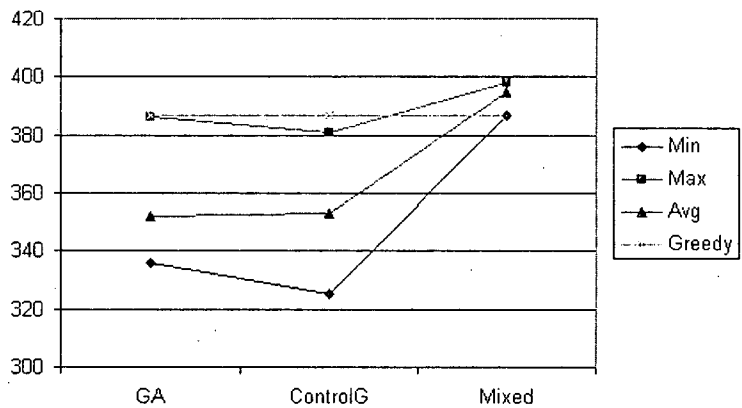


Figure 3: Results of the algorithms with 20 workers

6 Conclusions and future work

We solve this human resource allocation problem with a proper Greedy algorithm and with two genetic algorithms with different structures. The Greedy+GA algorithm depending on the cardinality of the workers has similar or better result than the Greedy result.
As future work we propose to search the students' preferences every year and analyze how to form better teams and what kind of networks they form. The knowledge

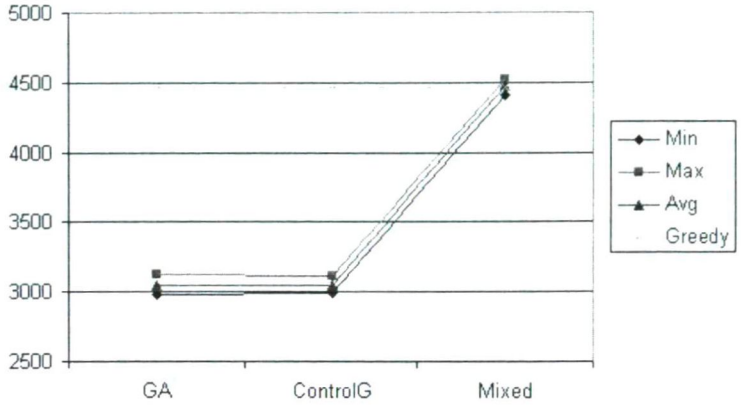


Figure 4: Results of the algorithms with 120 workers

of the students and some personality aptitudes are considered. Game-theory considerations are also a possible research topic for this problem, where we have an authority (the manager) and N players with different preferences. The players could lie if they are threatened to be eliminated. We can define types of individuals from the scores they give others if they do not lie. If somebody gives only high marks to the others, he may be exasperated about his job and hope to be chosen in a group. However, this idea would take us in the direction of psychology.

References

[1] Nichi . I., Avram-Nichi R. *On the Paradigm of Collaborative Support Systems*, In Proceedings of the Collaborative Support Systems in Business and Education, Academy of Economic Studies, Cluj Napoca, pages 274-292, 2005.

[2] Fábián, Cs. B. Private conversations 2003-2006.

[3] Darwin, C. *On the Origin of Species* John Murray, London, 1859.

[4] Holland, J.H. *Adaptation in Natural and Artificial Systems* Ann Arbour, University of Michigan Press, 1975.

[5] Jong, K.A.D. *An analysis of the behavior of a class of genetic adaptive adaptive systems.* PhD thesis, University of Michigan, 1975.

[6] Jong, K.A.D. A genetic-based global function optimization technique Technical Report, No.80-2, University of Pittsburgh, 1980.

[7] Jong, K.A.D. On using genetic algorithms to search program spaces. In Proceedings of the 2nd International Conference on *Genetic Algorithms and their Applications*, pages 210-216, Hillsdale, NJ, 1987.

[8] Hanli Wang et. al. Multi-objective hierarchical genetic algorithm for interpretable fuzzy rule-based knowledge extraction. *Fuzzy sets and systems*, 149:149–186, 2005.

[9] K. Meffert, N. Rotstan, Java Genetic Algorithms Package, http://jgap.sourceforge.net/

[10] A. Mescauscas, The traveling salesman problem http://jgap.sourceforge.net/doc/salesman/TravellingSalesman.html

[11] Illyés, L., Balanced student groups forming for university projects. In Proceedings of the 8th International Conference on *Informatics in Economy*, pages 554-559, Academy of Economic Studies, Bucharest, 2007

Acta Cybernetica 19 (2009) 105–123.

Adaptive Scheduling Solution for Grid Meta-Brokering*

Attila Kertész[†] József Dániel Dombi[‡] and József Dombi[‡]

Abstract

The nearly optimal, interoperable utilization of various grid resources play an important role in the world of grids. Though well-designed, evaluated and widely used resource brokers have been developed, these existing solutions still cannot cope with the high uncertainty ruling current grid systems. To ease the simultaneous utilization of different middleware systems, researchers need to revise current solutions. In this paper we propose advanced scheduling techniques with a weighted fitness function for an adaptive Meta-Brokering Grid Service, which enables a higher level utilization of the existing grid brokers. We also set up a grid simulation environment to demonstrate the efficiency of the proposed meta-level scheduling solution. The presented evaluation results show that the proposed novel scheduling technique in the meta-brokering context delivers better performance.

Keywords: grid computing, meta-brokering, scheduling, grid service, random number generator function

1 Introduction

Ten years ago a new computing infrastructure called the Grid was born. Ian Foster et. al. made this technology immortal by publishing the bible of the Grid [1] in 1998. Grid Computing has become an independent research field since then: currently grids are targeted by many world-wide projects. A decade is a long time: though the initial goal of grids to serve various scientific communities by providing a robust hardware and software environment is still unchanged, different middleware solutions have been developed (Globus Toolkit [2], EGEE [3], UNICORE [4], etc.). The realizations of these grid middleware systems formed production grids that are mature enough to serve scientists having computation and data intensive applications. Nowadays research directions are focusing on user needs: more efficient utilization and interoperability play the key roles. To solve these problems

*This work was supported by the FP7 Network of Excellence S-Cube funded by the European Commission (Contract FP7/2007-2013).

[†]MTA SZTAKI, University of Szeged E-mail: keratt@inf.u-szeged.hu

[‡]University of Szeged E-mail: {dombijd,dombi}@inf.u-szeged.hu

grid researchers have two options: as a member of a middleware developer group they can come up with new ideas or newly identified requirements and go through the long process of designing, standardizing and implementing the new feature, then waiting for the next release containing the solution. Researchers sitting on the other side or unwilling to wait for years for the new release, need to rely on the currently available interfaces of the middleware components and use advanced techniques of other related research fields (peer-to-peer, web computing, artificial intelligence, etc.). We have chosen the second option to improve grid resource utilization with an interoperable resource management service.

Since the management and advantageous utilization of highly dynamic grid resources cannot be handled by the users themselves, various grid resource management tools have been developed, supporting different grids. User requirements created certain properties that resource managers have learned to support. This development is still continuing, and users already need to stress themselves to distinguish brokers and to migrate their applications, when they move to a different grid. Interoperability problems and multi-broker utilization have emerged the need for higher level brokering solutions. The meta-brokering approach means a higher level resource management by enabling automatic and simultaneous utilization of grid brokers. Scheduling at this level requires sophisticated approaches, because high uncertainty presents at all stages of grid resource management. Despite these difficulties, this work addresses the resource management layer of middleware systems and proposes an enhanced scheduling technique to improve grid utilization in a high-level brokering service.

In the following sections of this paper we are focusing on a meta-brokering solution for grid resource management and present an adaptive scheduling technique that targets better scheduling in global grids. In Section 2 we introduce meta-brokering in grids, in Section 3 we describe our proposed scheduling solution, and in Section 4 we present our simulation architecture and the evaluation of our proposed solution. Finally, in Section 5 we gather the related research directions and Section 6 concludes the paper.

2 The need for grid meta-brokering

Heterogeneity appears not only in the fabric layer of grids, but also in the middleware. Even components and services of the same middleware may support different ways for accessing them. After a point this variety slows down grid development, and makes grid systems unmanageable. Most grid middleware systems have fixed interfaces to access their components and propagate information flow. In case of resource management most of the middleware systems provide access to static properties (number of CPUs, size of memory, etc.) and some also give dynamic ones (number of waiting jobs, expected response time), but this data is usually outdated due to timely periodic refreshing. As a result we can state that there is a high uncertainty in current grids, which is not likely to change soon. Though the first de facto middleware, the Globus Toolkit [2], did not have a resource broker that au-

tomates resource selection, the currently used middleware systems have built-in or supporting brokers [8]. The development of different brokers and grids has started a separation process in the research and user community, too. Therefore one of the major problems of current grids is grid interoperability. Focusing on the resource management layer of grids an obvious solution would be to interconnect brokers to create interoperability. Unfortunately current brokers do not have a common protocol and uniform interface for intercommunication, though the OGF-GSA [10] started to work on this issue. Once they standardize a solution we still would need to wait till all the brokers implement it in order to establish interoperability. In order to achieve this goal in a short term we have chosen to interconnect brokers by a high-level resource manager: we introduced meta-brokering (first proposed in [7]) that means a higher level utilization of the existing, widely used and reliable resource brokers. Since most of the users have certificates to access more Grids, they are facing the problem of grid selection: which grid, which broker should I choose for my specific application? Just like users needed resource brokers to choose proper resources within a grid, now they need a meta-brokering service to decide, which broker (or grid) is the best for them and also to hide the differences of utilizing them. In this way the meta-brokering approach solves the grid interoperability problem at the level of resource management by providing a uniform interface for the users of all the grids they have access to.

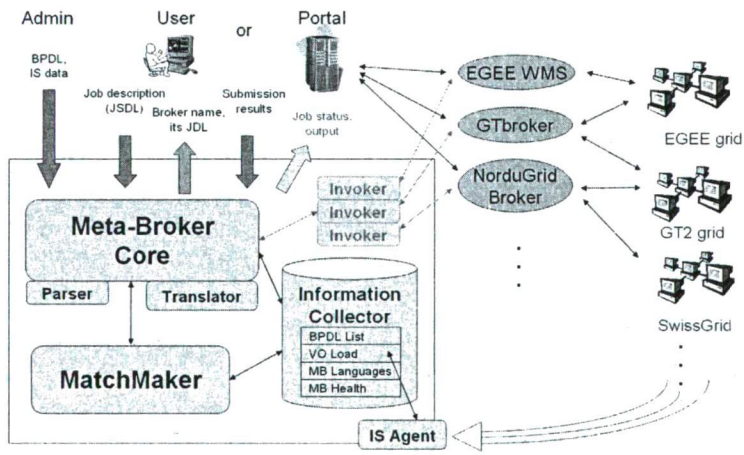


Figure 1: Components of the Grid Meta-Broker Service

Figure 1 introduces the revised architecture of the Grid Meta-Broker Service that enables the users to access resources of different grids through their own brokers. In this way, this higher level tool matches resource brokers to user requests. The system is implemented as a web-service that is independent from middleware-specific components. The provided services can be reached through WSDL (Web Services Description Language). In the following we give a brief summary of its com-

ponents and their operation. As the JSDL (Job Submission Description Language) standard [5] proposed by OGF (Open Grid Forum [9]) is general enough to describe jobs of different grids and brokers, we have chosen this to be the job description language of the Meta-Broker. The Translator components of the Meta-Broker are responsible for transforming the resource specification defined by the user to the language of the appropriate resource broker that the Meta-Broker selects to use for a given job. Regarding all the various job specification formats used by different grid middleware systems not all job attributes can be expressed in each document. Furthermore we revealed some useful scheduling-related attributes that are also missing from JSDL. To overcome these limitations we specified MBSDL (Meta-Broker Scheduling Description Language [6]). The main attribute categories are: middleware constraints, scheduling policies and QoS requirements. This schema can be used to extend JSDL with scheduling-related attributes. Besides describing user jobs, we also need to describe resource brokers in order to differentiate and manage them. These brokers have various features for supporting different user needs. These needs should be able to be expressed in the users JSDL, and identified by the Meta-Broker for each corresponding broker. Therefore we proposed an extendable BPDL (Broker Property Description Language [6]) – similar to the purpose of JSDL –, to express metadata about brokers. The common subset of the individual broker properties is stored here: the supported middleware, job types, certificates, job descriptions, interfaces, monitoring features and dynamic performance data. The scheduling-related ones are stored in MBSDL: fault tolerant features (checkpointing, rescheduling, replication), agreement support, scheduling policies (ranking by resource attributes) and QoS properties (e.g. advance reservation, co-allocation, email notification). The union of these properties forms a complete broker description document that can be filled out and regularly updated for each utilized resource broker. These two kinds of data formats are used by the Meta-Broker: JSDL is used by the users to specify jobs and the BPDL (Broker Property Description Language) by administrators to specify brokers – both parties can use MBSDL to extend their descriptions.

The Information Collector (IC) component of the Meta-Broker stores the data of the reachable brokers and historical data of the previous submissions. This information shows whether the chosen broker is available, or how reliable its services are. During broker utilization the successful submissions and failures are tracked, and regarding these events a rank is modified for each special attribute in the BPDL of the appropriate broker (these attributes were listed above). In this way, the BPDL documents represent and store the dynamic states of the brokers. All data is stored in XML, and advanced XML-serialization techniques are used by the IC. The load of the resources behind the brokers is also taken into account to help the Matchmaker to select the proper environment for the actual job. When too many similar jobs are needed to be handled by the Meta-Broker an eager matchmaking may flood a broker and its grid. That is the main reason why load balancing is an important issue. In order to cope with this problem, there is an IS (Information System) Agent component reporting to the Information Collector, which regularly checks the load of the underlying grids of each connected resource broker, and store this data. This

tool is implemented as separate web-service connected to the Information System of the grids behind the utilized brokers. With the additional information provided by this agent the matchmaking process can adapt to the load of the utilized grids. Finally, the actual state (load, configurations) of the Meta-Broker is also stored here, and it can also be queried by users. The continuous monitoring of grid load and broker performances makes this grid service self-adaptive.

The previously introduced languages are used for matching the user requests to the description of the interconnected brokers: which is the role of the Matchmaker component. The JSDL contains the user request (this supposed to be an exact specification of the user's job) using the extended attributes, while the interconnected brokers are described by their BPDL documents. The default matchmaking process consists of the following steps to find the fittest broker:

- The Matchmaker compares the JSDL of the actual job to the BPDL of the registered resource brokers. First the job requirement attributes are matched against the broker properties stored in their BPDLs: this selection determines a group of brokers that are able to submit the job. This phase consists of two steps: first the brokers are filtered by all the requirements stated in the JSDL. When none of the brokers can fulfill the request, another filtering process will be started with minimal requirements (those ones are kept which are real necessary for job execution). If the available brokers still can not accept the job, it will be rejected.
- In the second phase the previous submissions of the brokers and the load of the underlying grids are taken into account: The MatchMaker counts a rank for each of the remaining brokers. This rank is calculated from the load that the IS Agent regularly updates, and from the job completion rate that is updated in the PerformanceMetrics field of the BPDL for each broker. When all the ranks are counted, the list of the brokers is ordered by these ranks.
- Finally the first broker of the priority list is selected for submission.

3 Adaptive Scheduling for meta-brokering

In the previous section we introduced the Grid Meta-Broker and shown how the default matchmaking is carried out. The main goal of this paper is to enhance the scheduling part of this matchmaking process. To achieve this, we have created a Decision Maker component and inserted it into the MatchMaker component of the Meta-Broker (see Figure 1). The first part of the matchmaking is unchanged: the list of the available brokers is filtered according to the requirements of the actual job read from its JSDL. Then the list of the remaining brokers along with their performance data and background grid load are sent to the Decision Maker in order to determine the fittest broker for the actual job. The scheduling techniques and the process are described in the following paragraphs.

The Decision Maker uses a random number generator, and we chose a JAVA solution, which generates pseudorandom numbers. This generator produces exactly

the same sequence of random numbers for each execution with the same initial value. This initial value is called the seed. The JAVA random number generator class uses uniform distribution and 48-bit seed, which is modified by a linear congruential formula[11]. The default seed is the current time in milliseconds since 1970. We also developed a unique random number generator, which generates random numbers with a given distribution. We called this algorithm as generator function. In our case we defined a score value for each broker, and we created the distribution based on the score value. For example the broker which has the highest score number has the highest probability to be chosen. In this algorithm the inputs are the broker id and the broker score, which are integer numbers (see Table 1).

Table 1: Inputs of the algorithm

BrokerID	Score
3	2
4	3
5	1
6	2

The next step is to choose a broker and put it into a temporary array: the cardinality is determined by the score value (see Table 2). After the temporary array is filled, we shuffle this array and choose an array element using the JAVA random generator. In the example shown in Table 3 the generator function chose the broker with id 4.

Table 2: Elements in the temporary array

Broker ID	3	3	4	4	4	5	6	6
Array ID	1	2	3	4	5	6	7	8

Table 3: Shuffled temporary array

Broker ID	4	3	6	3	4	4	5	6
Array ID	1	2	3	4	5	6	7	8

Java Random generator: **5**

To improve the scheduling performance of the Meta-Broker we need to send the job to the broker that best fits the requirements and executes the job without failures with the shortest execution time. Every broker has three properties that

the algorithm can rely on: the successful counter, the failure counter and the load counter.

- The successful counter represents the number of jobs which had finished without any errors.
- The failure counter shows the number of failed jobs.
- The load counter indicates the actual load of the grid behind the broker (in percentage).

We have developed four different kinds of decision algorithms. The trivial algorithm uses only a random number generator to select a broker. The other three algorithms take into account the previously mentioned broker properties. These algorithms define a score number for each broker and use the generator function to select one. To calculate the score value we build a weighted sum of the evaluated properties. This number is always an integer number. Furthermore, the second and third decision algorithms take into account the maximum value of the failure and load counter. This means that we extract the maximum value of the properties before multiplying them with the weight. The generator function of the third algorithm chooses a broker which score number is not smaller than the half of the highest score value.

After testing different kinds of weighted systems, we conclude that the most useful weights are shown in Table 4) that represent the weights of the used decision algorithms.

Table 4: The weights of the decision makers

Decision Maker	Success_weight	Failed_weight	Load_weight
Decision I.	3	0.5	1
Decision II.	4	4	4
Decision III.	4	4	4

During the utilization of the Meta-Broker, the first two broker properties (successful and failure counter) are incremented through a feedback method that the simulator (or a user or portal in real world cases) calls after the job submission is finished. The third property, the load value, is queried by the Meta-Broker from an information provider (Information System) of a Grid. During simulation this data is saved to a database by the Broker entities of the simulator (described later and shown in Figure 2). This means by the time we start the evaluation and till we do not receive feedback from finished jobs the algorithms can only rely on the background load of the grids. To further enhance the scheduling we developed a training process that can be executed before the simulation in order to initialize the first and second properties. This process sends a small number of jobs with various properties to the brokers and set the successful and failed jobs number at

the BPDLs of the brokers. With this additional training method we expect shorter execution times by selecting more reliably brokers.

4 Evaluation

In order to evaluate our proposed scheduling solution, we have created a general simulation environment, in which all the related grid resource management entities can be simulated and coordinated. The GridSim toolkit [12] is a fully extendable, widely used and accepted grid simulation tool – these are the main reasons why we have chosen this toolkit for our simulations. It can be used for evaluating VO-based resource allocation, workflow scheduling, and dynamic resource provisioning techniques in global grids. It supports modeling and simulation of heterogeneous grid resources, users, applications, brokers and schedulers in a grid computing environment. It provides primitives for the creation of jobs (called gridlets), mapping these jobs to resources and their management, therefore resource schedulers can be simulated to study scheduling algorithms. GridSim provides a multilayered design architecture based on SimJava [13], a general purpose discrete-event simulation package implemented in Java. It is used for handling the interaction or events among GridSim components. All components in GridSim communicate with each other through message passing operations defined by SimJava.

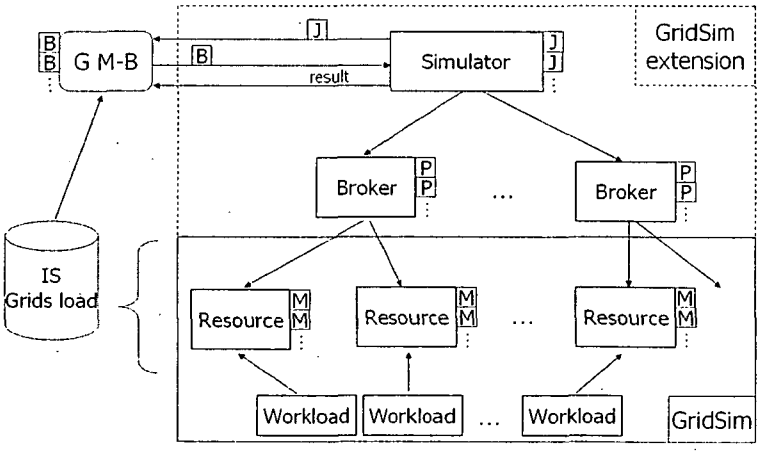


Figure 2: Meta-Brokering simulation environment based on GridSim

Our general simulation architecture can be seen in Figure 2. On the bottom-right part we can see the GridSim components used for the simulated grid systems. Resources can be defined with different grid-types. Resources consist of more machines, to which workloads can be set. On top of this simulated grid infrastructure we can set up brokers. The Broker and Simulator entities have been developed by us in order to enable the simulation of meta-brokering. Brokers are extended

GridUser entities:

- they can be connected to one or more resources;
- different properties can be set to these brokers (agreement handling, co-allocation, advance reservation, etc.);
- some properties can be marked as unreliable;
- various scheduling policies can be defined (pre-defined ones: rnd – random resource selection, fcpu – resources having more free cpus or less waiting jobs are selected, nfailed – resources having less machine failures are selected);
- generally resubmission is used, when a job fails due to resource failure;
- finally they report to the IS (Information System) Grid load database by calling the feedback method of the Meta-Broker with the results of the job submissions (this database has a similar purpose as a grid Information System).

The Simulator is an extended GridSim entity:

- it can generate a requested number of gridlets (jobs) with different properties, start and run time (length);
- it is connected to the created brokers and able to submit jobs to them;
- the default job distribution is the random broker selection (though at least the middleware types are taken into account);
- in case of job failures a different broker is selected for the actual job;
- it is also connected to the Grid Meta-Broker through its web service interface and able to call its matchmaking service for broker selection.

4.1 Preliminary testing phase

Table 5 shows the details of the preliminary evaluation environment. 10 brokers can be used in this simulation environment. The second column denotes the scheduling policies used by the brokers: fcpu means the jobs are scheduled to the resource with the most free cpus, nfail means those resources are selected that have less machine failures, and rnd means randomized resource selection. The third column shows the capabilities/properties (eg: coallocation, checkpointing, ...) of the brokers: three properties are used in this environment, subscript F means unreliability, a broker having such a property may fail to execute a job with the requested service with a probablity of 0.5. The fourth column contains the number of resources utilized by a broker, while the fifth column contains the number of background jobs submitted to the broker (SDSC BLUE workload logs taken from the Parallel Workloads Archive [14]) during the evaluation timeframe.

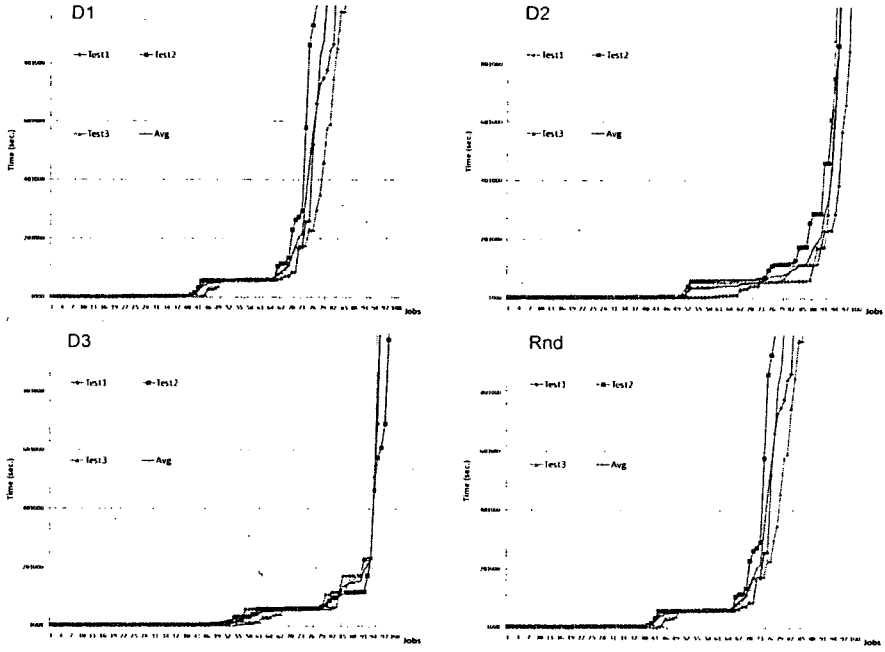


Figure 3: Diagrams of the preliminary evaluation for each algorithm

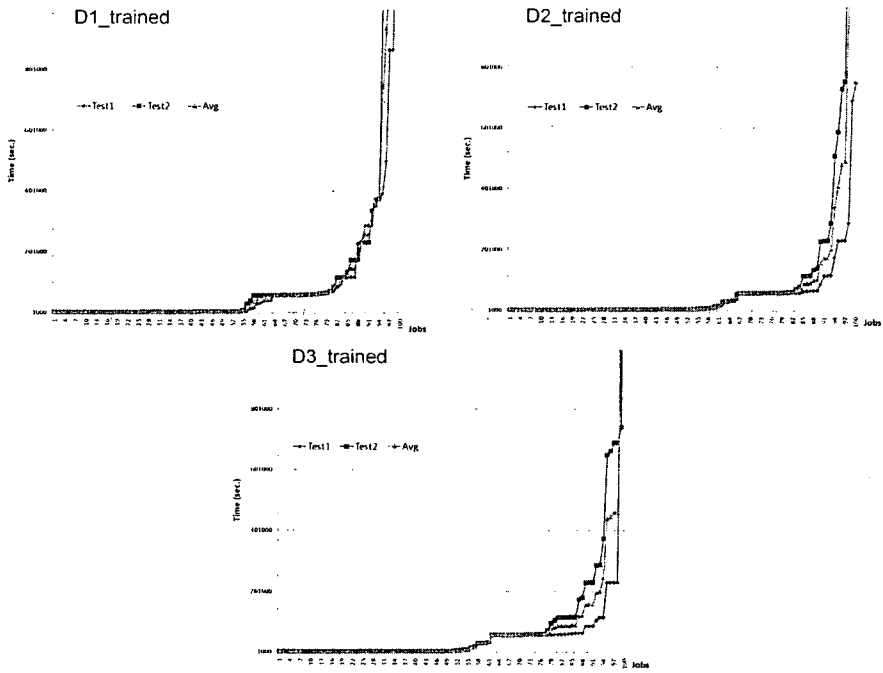


Figure 4: Diagrams of the preliminary evaluation for each algorithm with training phases

Table 5: Preliminary evaluation setup.

Broker	Scheduling	Properties	Resources	Workload
1.	fcpu	A	8	20*8
2.	fcpu	B	8	20*8
3.	fcpu	C	8	20*8
4.	fcpu	A_F	8	20*8
5.	fcpu	B_F	8	20*8
6.	fcpu	C_F	8	20*8
7.	nfail	$A_F B$	10	20*10
8.	nfail	AC_F	10	20*10
9.	nfail	$B_F C$	10	20*10
10.	rnd	-	16	20*16

As shown in the table we utilized 10 brokers to execute our first experiment. In this case we submitted 100 jobs to the system, and measured the makespan of all the jobs (time elapsed from submission till the successful finishing, including waiting time in the queue of the resources and resubmissions on failures). Out of the 100 jobs 40 had no special property (this means all the brokers can successfully execute them), for the rest of the jobs the three properties were distributed equally: 20 jobs had property A, 20 had B and 20 had C. Each resource of the simulated grids has been utilized by 20 background jobs (workload) with different submission times according to the distribution defined by the SDSC BLUE workload logs.

Figure 3 shows the detailed evaluation runs with the scheduling algorithms Decision 1 (D1), 2 (D2), 3 (D3) and without the use of the Meta-Broker (randomized broker selection – Rnd) respectively. Figure 4 shows the measured values with the three algorithms with training (we submitted 10 jobs to each broker to set their initial performance values). In Figure 5 we can see the averages of the tests with the different algorithms. This illustrates best the differences of the simulations with and without the use of the Meta-Broker.

After we have seen the diagrams of the preliminary evaluations we can state that all the proposed scheduling algorithms (D1, D2 and D3) provide shorter execution times than the random broker selection. In the main evaluation phases our goal was to set up a more realistic environment and to experience with a higher number of jobs.

4.2 Main testing phase

Table 6 shows the evaluation environment used in the main evaluation. The simulation setup was derived from real-life production grids: current grids and brokers support ony a few special properties: we used four. To determine the (proportional) number of resources in our simulated grids we compared the sizes of current production grids (EGEE VOs, DAS3, NGS, Grid5000, OSG, ...). We used the same

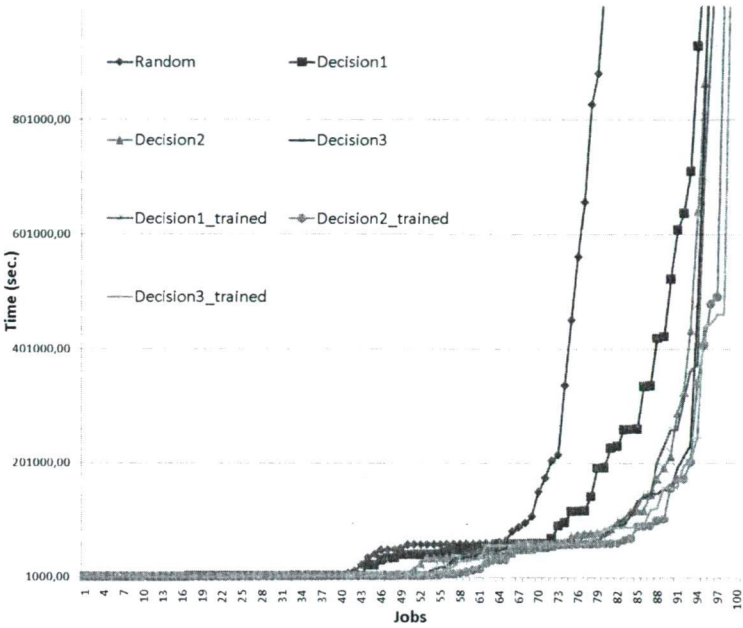


Figure 5: Summary diagram of the preliminary evaluation

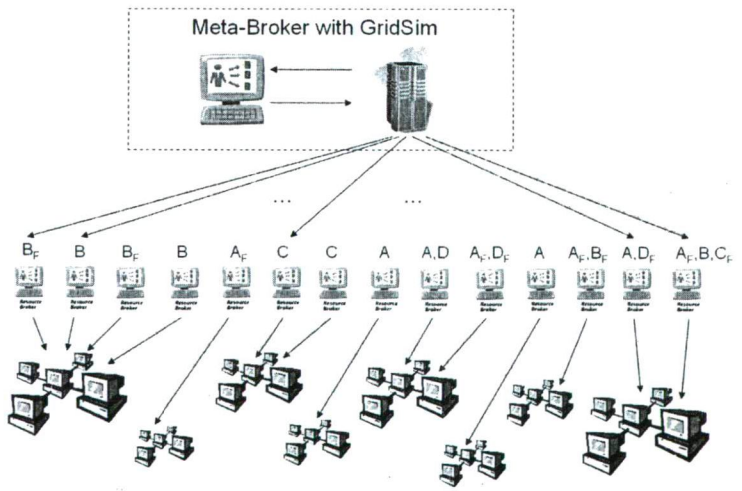


Figure 6: Simulation in the main evaluation environment

Table 6: Main evaluation setup.

Broker	Scheduling	Properties	Resources	Workload
1.	fcpu	A	6	50*6
2.	fcpu	A_F	8	50*8
3.	fcpu	A	12	50*12
4.	fcpu	B	10	50*10
5.	fcpu	B_F	10	50*10
6.	fcpu	B	12	50*12
7.	fcpu	B_F	12	50*12
8.	fcpu	C	4	50*4
9.	fcpu	C	4	50*4
10.	fcpu	$A_F D$	8	50*8
11.	fcpu	AD	10	50*10
12.	fcpu	AD_F	8	50*8
13.	fcpu	AB_F	6	50*6
14.	fcpu	ABC_F	10	50*10

notations in this table as before.

In the main evaluation we utilized 14 brokers. In this case we submitted 1000 jobs to the system, and again measured the makespan of all the jobs. Out of the 1000 jobs 100 had no special property, for the rest of the jobs the four properties were distributed in the following way: 30 jobs had property A, 30 had B, 20 had C and 10 had D. The workload logs contained 50 jobs for each resource. In the training processes 100 jobs were submitted to each broker prior the evaluations to set the initial values. Figure 6 shows the graphical representation of the simulation environment.

In the first phase of the main evaluation the simulator submitted all the jobs at once, just like in the preliminary evaluation. The results for this phase can be seen in Figure 7.

In the first phase we could not exploit all the features of the algorithms, because we submitted all the jobs at once and the performance data of the brokers were not updated early enough for the matchmaking process. To avoid this, in the last phase of the main evaluation we submitted the jobs periodically: 1/3 of the jobs were submitted in the beginning, then the simulator waited for 200 jobs to finish and update the performances of the brokers. After this the simulator submitted again 1/3 of all the jobs and waited for 300 more to finish. Finally the rest of the jobs (1/3 again) were submitted. In this way the broker performance results could be used by the scheduling algorithms. Figure 8 shows the results of the last evaluation phase. Here we can see that the runs with training could not make too much advantage of the trained values, because the feedback of the first submission period compensates the lack of training.

Figure 9 displays the summary of the different evaluation phases. The depicted

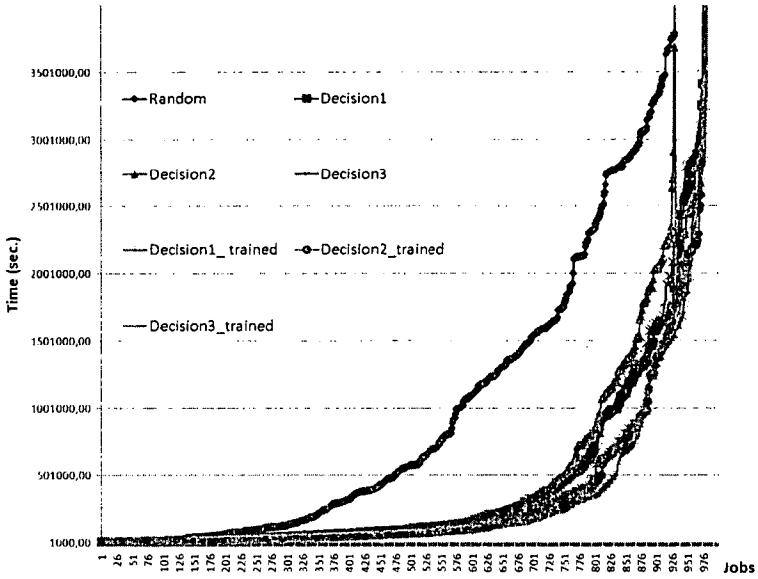


Figure 7: Diagram of the first phase of the main evaluation

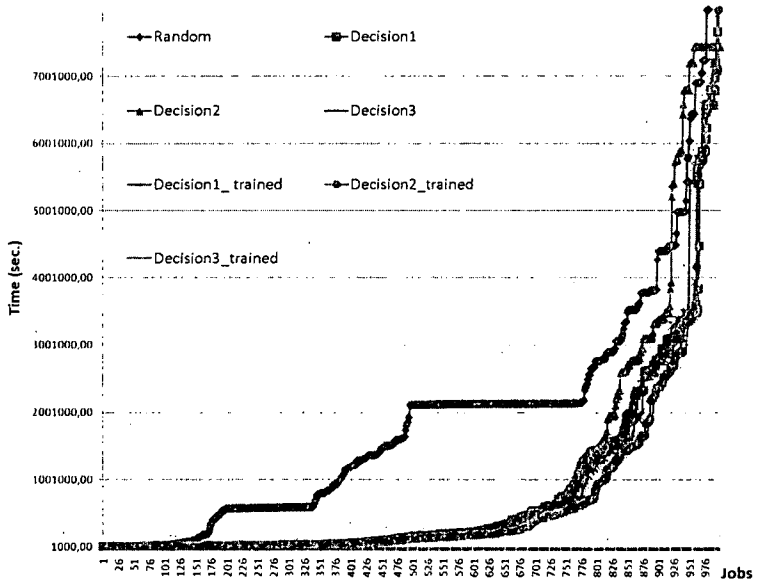


Figure 8: Diagram of the second phase of the main evaluation

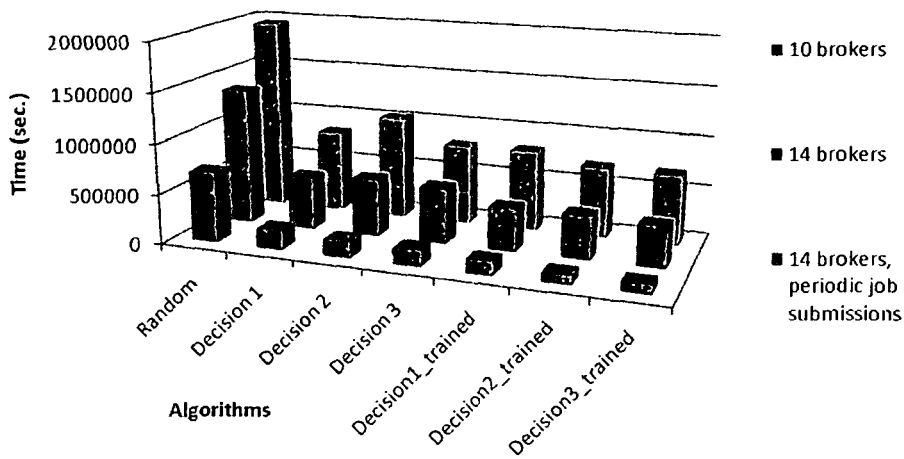


Figure 9: Summary of the evaluation results

columns show the average values of each evaluation runs with the same parameters. The results clearly show that the more intelligence (more sophisticated methods) we put into the system the higher performance we gain. The most advanced version of our proposed meta-brokering solution is the Decision Maker with the algorithm called Decision3 with training. Once the number of brokers and job properties will be high enough to set up this Grid Meta-Broker Service for inter-connecting several Grids, with the presented scheduling algorithms our service will be ready to serve thousands of users even under high uncertainty.

5 Related work

Meta-brokering means a higher level solution that brokers user requests among various grid domains. One of these promising approaches aims at enabling communication among existing resource brokers. The GSA-RG of OGF [10] is currently working on a project enabling grid scheduler interaction. They try to define common protocol and interface among schedulers enabling inter-grid usage. In this work they propose a Scheduling Description Language to extend the currently available job description language solutions. This work is still in progress, up to now only a partial SDL schema has been created. The meta-scheduling project in LA Grid [15] aims to support grid applications with resources located and managed in different domains. They define broker instances with a set of functional modules: connection management, resource management, job management and notification management. These modules implement the APIs and protocols used in LA Grid through web services. Each broker instance collects resource information from its neighbors and save the information in its resource repository or in-core memory. The resource information is distributed in the Grid and each instance will have a

view of all resources. The Koala grid scheduler [16] was designed to work on DAS-2 interacting with Globus [2] middleware services with the main features of data and processor co-allocation; lately it is being extended to support DAS-3 and Grid'5000. To inter-connect different grids, they have also decided to use inter-broker communication. Their policy is to use a remote grid only if the local one is saturated. In an ongoing experiment they use a so-called delegated matchmaking (DMM), where Koala instances delegate resource information in a peer-2-peer manner. Gridway introduces a Scheduling Architectures Taxonomy [17], where they describe a Multiple Grid Infrastructure. It consists of different categories, we are interested in the Multiple Meta-Scheduler Layers, where Gridway instances can communicate and interact through grid gateways. These instances can access resources belonging to different administrative domains (grids/VOs). They pass user requests to another domain when the current is overloaded – this approach follows the same idea as the previously introduced DMM. Gridway is also based on Globus, and they are experimenting with GT4 and gLite [3]. Comparing the previous approaches, we can see that all of them use a new method to expand current grid resource management boundaries. Meta-brokering appears in a sense that different domains are being examined as a whole, but they rather delegate resource information among domains, broker instances or gateways. Usually the local domain has preference, and when a job is passed to somewhere else, the result should be passed back to the initial point. Regarding multi-grid usage, the existing grids are very strict and conservative in the sense that they are very reluctant to introduce any modification that is coming from research or from other grid initiatives. Hence the solutions aiming at inter-connecting the existing brokers through common interfaces require a long standardization procedure before it will be accepted and adapted by the various grid communities. On the other hand the advantage of our proposed meta-brokering concept is that it does not require any modification of the existing grid schedulers, since it utilizes and delegates broker information by reaching them through their current interfaces. The HPC-Europa Project researchers also considered taking steps towards meta-brokering [18]; currently we have an ongoing work together with them to define a common meta-brokering model.

6 Conclusions

The Grid Meta-Broker itself is a standalone Web-Service that can serve both users and grid portals. The presented enhanced, adaptive scheduling solution with this Meta-Broker enables a higher level, interoperable brokering by utilizing existing resource brokers of different grid middleware. It gathers and utilizes meta-data about existing widely used brokers from various grid systems to establish an adaptive meta-brokering service. We have developed a new scheduling component for this Meta-Broker called Decision Maker that uses weighted functions with random generation to select a good performing broker to user jobs even under high uncertainty. We have evaluated the presented algorithms in a simulation environment based on GridSim with real workload samples. The presented evaluation results

affirm our expected utilization gains: the enhanced scheduling provided by the Decision Maker enables better adaptation and results in a more efficient job execution.

References

[1] Foster, I. and Kesselman, C. *Computational Grids, The Grid: Blueprint for a New Computing Infrastructure*, pp. 15–52, Morgan Kaufmann, 1998.

[2] Foster, I. and Kesselman, C. *The Globus project: A status report*, pp. 4–18, In Proc. of the Heterogeneous Computing Workshop, IEEE Computer Society Press, 1998.

[3] *EGEE middleware technical website*, `http://egee-technical.web.cern.ch/egee-technical`, September 2008.

[4] Erwin, D. W. and Snelling, D. F. *UNICORE: A Grid Computing Environment*, pp. 825–834, In Lecture Notes in Computer Science, volume 2150, Springer, 2001.

[5] *Job Submission Description Language (JSDL)*, `http://www.ggf.org/documents/-GFD.56.pdf`, September 2008.

[6] Kertész, A., Kacsuk, P., Rodero, I., Guim, F. and Corbalan, J. *Meta-Brokering requirements and research directions in state-of-the-art Grid Resource Management*, Technical report, TR-0116, Institute on Resource Management and Scheduling, CoreGRID – Network of Excellence, November 2007.

[7] Kertész, A. and Kacsuk, P. *Grid Meta-Broker Architecture: Towards an Interoperable Grid Resource Brokering Service*, pp. 112–116, CoreGRID Workshop on Grid Middleware in conjunction with Euro-Par 2006, Dresden, Germany, LNCS, Vol. 4375, 2007.

[8] Kertész, A. and Kacsuk, P. *A Taxonomy of Grid Resource Brokers*, pp. 201–210, 6th Austrian-Hungarian Workshop on Distributed and Parallel Systems (DAPSYS 2006), Springer US, 2007.

[9] *Open Grid Forum (OGF) website*, http://www.ogf.org, September 2008.

[10] *OGF Grid Scheduling Architecture Research Group*, `https://forge.gridforum.org/sf/pro-jects/gsa-rg`, September 2008.

[11] Knuth, Donald E. *The Art of Computer Programming Volume 2.*, Section 3.2.1. Addison-Wesley Professional, 1997.

[12] Buyya, B. and Murshed, M. *GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing*, pp. 1175–1220, Concurrency and Computation: Practice and Experience, Volume 14, Issue 13-15, 2002.

[13] Howell, F. and McNab, R. *SimJava: A discrete event simulation library for Java*, In Proc. of the International Conference on Web-Based Modeling and Simulation, San Diego, USA, 1998.

[14] *Parallel Workloads Archive website*, `http://www.cs.huji.ac.il/labs/parallel/workload`, September 2008.

[15] Rodero, I., Guim, F. and Corbalan, J., Fong, L.L., Liu, Y.G. and Sadjadi, S.M. *Looking for an Evolution of Grid Scheduling: Meta-brokering*, Coregrid Workshop in Grid Middleware'07, Dresden, Germany, June 2007.

[16] Iosup, A., Epema, D.H.J., Tannenbaum, T., Farrellee, M. and Livny, M. *Inter-Operating Grids through Delegated MatchMaking*, In proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC07), Reno, Nevada, November 2007.

[17] Vazquez, T., Huedo, E., Montero, R. S. and Llorente, I. M. *Evaluation of a Utility Computing Model Based on the Federation of Grid Infrastructures*, pp. 372–381, Euro-Par 2007, August 28, 2007.

[18] *The HPC-Europa Project website*, http://www.hpc-europa.org, September 2008.

Acta Cybernetica **19** (2009) 125–133.

Parameter Learning Online Algorithm for Multiprocessor Scheduling with Rejection*

Tamás Németh† and Csanád Imreh†

Abstract

In multiprocessor scheduling with rejection the jobs are characterized by a processing time and a penalty and it is possible to reject the jobs. The goal is to minimize the makespan of the schedule for the accepted jobs plus the sum of the penalties of the rejected jobs. In this paper we present a new online algorithm for the problem. Our algorithm is a parameter learning extension of the total reject penalty algorithm. The efficiency of the algorithm is investigated by an experimental analysis.

Keywords: online algorithms, scheduling, experimental analysis

1 Introduction

In this paper we develop a new algorithm for the solution of the online scheduling with rejection problem on identical machines. The algorithm is based on the idea of learning the parameter of the Reject Total Penalty (RTP) algorithm. We measure the efficiency of the new algorithm by an experimental analysis.

The problem of scheduling with rejection is defined in [2]. In this model, it is possible to reject the jobs. The jobs are characterized by a processing time and a penalty. The goal is to minimize the makespan of the schedule for the accepted jobs plus the sum of the penalties of the rejected jobs. In the online case a 2.618-competitive algorithm is given for arbitrary number of machines. This algorithm is called Reject Total Penalty (RTP). One basic idea in scheduling with rejection is to compare the penalty and the load (processing time divided by the number of machines) of the job, and reject the job in the case when the penalty is smaller. This greedy algorithm can make a bad decision when the number of machines is large and this makes possible to appear large jobs with small loads. RTP handles these jobs more carefully. We give the detailed definition in the next section. In [2] a further, 1.618-competitive algorithm is presented in the case of 2 machines. Matching lower bounds are also given. In the offline case an FPTAS is presented for fixed number

*This research has been supported by the Hungarian National Foundation for Scientific Research, Grant F048587.

†Institute of Informatics, University of Szeged E-mail: {tnemeth,cimreh}@inf.u-szeged.hu

of machines, and a PTAS in the case where the number of machines is part of the input. The preemptive version of online scheduling with rejection is studied in [14], a generalized version of the reject total penalty algorithm is analyzed, and it is proved that this generalized algorithm is 2.387-competitive for arbitrary number of machines. A general lower bound of 2.124, and a lower bound of 2.33 for the class of obliviously scheduling algorithms (the accepted jobs are scheduled without knowledge of the rejection penalties) are also proved. In [7] the offline scheduling problem with rejection is investigated in some more complex machine models. In [8] an FPTAS is given for scheduling with rejection on related parallel machines. A further extension of the problem where the machines have cost and the algorithm has to purchase the machines before using them is investigated in [6] and [13]. A general machine scheduling problem on two sets of machines which generalizes the problem of scheduling with rejection is presented in [11].

Typically, the quality of an online algorithm is judged using competitive analysis. An online scheduling algorithm is C-competitive if the algorithm cost is never more than C times the optimal cost. On can find many details about competitive analysis in [4], [9] and [12]. Considering the competitive ratio the problem of scheduling with rejection on identical machines is completely solved in the general case, where no further restrictions are given on the jobs, algorithm RTP is an optimal online algorithm in the sense that it achieves the smallest possible competitive ratio. On the other hand in some cases the algorithm which has the best competitive ratio does not work well in average cases or on real data sets. In the area of online scheduling algorithms only a few papers present experimental studies, such papers are, for example, the next ones: In [1] the algorithms for online multiprocessor scheduling to minimize the makespan are investigated. In [3] a multicriteria version of the scheduling problem is studied. In [5] online scheduling with release dates to minimize the weighted total completion time is investigated.

The paper is organized as follows. In the next section we introduce the basic notations and recall the most important results which are used in the paper. In Section 3 we present the developed parameter learning algorithm. Section 4 contains the description of the experimental analysis, and the evaluation of the results. In Section 5 we summarize the results and list some further open problems related to the paper.

2 Notations and preliminaries

In the problem considered there are m identical machines. Each job j has a processing time denoted by p_j and a penalty denoted by w_j. For an arbitrary list J of jobs and an algorithm A, we denote by $A(J)$ the cost of the schedule produced by algorithm A on list J.

As a subroutine we will use an online scheduling algorithm. Several algorithms are developed for the online scheduling problem on n identical machines (see the survey [15]), we will use the classical, greedy online scheduling algorithm LIST ([10]). This algorithm always schedules greedily the arriving job on the least loaded

machine.

Considering the problem of multiprocessor scheduling with rejection, it is a straightforward idea to reject the jobs where the penalty is smaller than the load. The following greedy algorithm gives this solution for the problem.

Algorithm Greedy

If $w_j \leq p_j/m$ is valid for job j then reject it otherwise accept and schedule it by LIST.

Unfortunately Greedy makes bad decisions in some cases. If only one job arrives with a large processing time M and penalty $M/m + \varepsilon$, then Greedy accepts and schedules it, and this shows that it is not better than m competitive. Therefore its competitive ratio is not constant.

In [2] an algorithm called RTP is defined which achieves a constant competitive ratio. It is a refined version of Greedy, it also rejects some large jobs with $w_j > p_j/m$, these jobs are collected in set R. We can define this algorithm as follows.

Algorithm RTP(α)

- 1. *Initialization.* Let $R := \emptyset$.
- 2. When job j arrives
  - (i) If $w_j \leq \frac{p_j}{m}$, then reject.
  - (ii) Let $r = \sum_{i \in R} w_i + w_j$. If $r \leq \alpha \cdot p_j$, then reject job j, and set $R = R \cup \{j\}$.
  - (iii) Otherwise, accept j and schedule it by LIST

In [2] the following statement is proved about the competitive ratio of RTP.

Proposition 1. *RTP is* $(3+\sqrt{(5)})/2$ *competitive, with parameter* $\alpha = (\sqrt{(5)}-1)/2$

In the same paper it is proved that no algorithm with better competitive ratio exists.

Proposition 2. *There exists no online algorithm that is* β*-competitive for some constant* $\beta < (3 + \sqrt{5})/2$ *and for all* m.

3 Parameter learning algorithm

Proposition 1 and Proposition 2 solves the problem of multiprocessor scheduling with rejection on identical machines for the general case as far as the competitive analysis is concerned. $RTP((\sqrt{5}-1)/2)$ is the best possible online algorithm for the solution of the problem. On the other hand sometimes the algorithms which have better competitive ratio show worst performance in average case on real or randomly generated inputs. In the area of online scheduling such example can be found in [1]

where the online algorithms for multiprocessor scheduling to minimize makespan are analysed by an experimental analysis and it is shown that the simple LIST algorithm has better performance than some of the more complicated algorithm with smaller competitive ratio.

In the case of scheduling with rejection the parameter $\alpha = (\sqrt{5}-1)/2$ is the value which minimizes the competitive ratio, thus it is a natural question whether using some other parameter can improve the average case. In this section we present a new algorithm PAROLE (Parameter Online Learning) which tries to learn the best parameter during its execution. The algorithm works in phases, after each phase it chooses a new parameter based on the known part of the input. First we define a frame algorithm which uses the selection of the new parameter as a subrutin, then we define the subrutin which finds the new parameter. We defined the phases by the number of arriving jobs, the phase is finished after 250 jobs.

Algorithm PAROLE (PHASE i)

- At the beginning of phase i, use algorithm CHOOSE to find a new parameter α_i.
- Perform $RTP(\alpha_i)$ on the arrived part of the input. Change set R.
- Use $RTP(\alpha_i)$ for the jobs arriving during the phase.

It is a straightforward idea to use the value α_i where the cost $RTP(\alpha_i)(I)$ is minimal for the known part of input. Unfortunately it seems to be difficult to find the optimal value of the parameter. $RTP(\alpha)(I)$ is neither monotone nor continuous function of α. It is a step function, but it may have many pieces. Our conjecture is that it is an NP-hard problem to find the optimal value of α, but it seems to be difficult to prove that. Therefore we used the following sampling algorithm to find the new value of the parameter. The algorithm uses the previous value of the parameter denoted by α^*.

Algorithm CHOOSE

- Generate one element from the intervals $[(i-1)/10, i/10]$ by uniform distribution for $i = 1, \ldots, 10$. Denote this set by S_1. Consider the value α from S_1 where $RTP(\alpha)$ has the smallest cost on I. Denote it by $\bar{\alpha}$.
- Generate one element from the intervals $[\alpha^* - i/100, \alpha^* - (i-1)/100]$, $[\alpha^* + (i-1)/100, \alpha^* + i/100]$, $[\bar{\alpha} - i/100, \bar{\alpha} - (i-1)/100]$, $[\bar{\alpha} + (i-1)/100, \bar{\alpha} + i/100]$ for $i = 1, \ldots 10$ by uniform distribution. Denote the set of the generated elements by S_2. Return the value α from $S_2 \cup \{\alpha^*\} \cup \{\bar{\alpha}\}$ where $RTP(\alpha)$ has the smallest cost on I.

The basic idea of CHOOSE is to select a good parameter value. We investigate the neighborhoods of two candidates, one is the previous value of the parameter and a further one is a new value $\bar{\alpha}$ which is the best among several candidates selected independently on the previous value of the parameter.

4 Experimental analysis

4.1 Description of the performed tests

To investigate the performance of the new parameter learning algorithm we performed the following experimental study. We have implemented the algorithms presented above and analysed their behavior on randomly generated test cases and on real data.

During the analysis we investigated the following algorithms.

- the Greedy algorithm
- $RTP(\alpha)$ with $\alpha = (\sqrt{5} - 1)/2$
- the parameter learning algorithm *PAROLE*.

To test the algorithms we considered the following classes of input. We used real data sets (Tests A and B) furthermore we used randomly generated inputs which were defined by the same distributions as in [1] and [3]. In each cases we used relatively large inputs in the tests, the reason is that for large inputs PAROLE has enough possibility to learn and use the best value of the parameter.

- Test A (real data): We collected the processing times of the tasks on the server www.szakoktatas.hu. The database contained around 100000 jobs. The average size of the jobs were 2597.1, the standard deviation was 20129.96. The smallest job had size 1, the largest had size 3134460. Therefore the jobs sizes followed the situation described in [1], there were very large jobs. On the server each process had a priority value which measured the importance of the task, this value is received by some properties of the jobs. The following properties were considered: the owner of the jobs (each user had a priority), the type of the job (the jobs are assigned to different classes by their type and each class has some priority value), some jobs had deadline (the deadline were also taken into account). We used a linear combination of these values to define the penalty. The average penalty was 272.83 the standard deviation was 229.32. The minimal penalty was 1, and the maximal was 2613.

- Test B (real data): We collected the processing times of the tasks on the server www.moravarosi.hu. The database contained around 200000 jobs and the jobs were larger than in the previous test. The average size of the jobs were 15608.96, the standard deviation was 120991.53. The smallest job had size 1, the largest had size 18839728. We used the priority to define the penalty in the same way as in Test A. We obtained larger penalties, the average penalty was 572.99, the standard deviation was 481.58. The minimal penalty was 1, the maximal one was 5487.

- Test C: In this case we generated the size of the jobs by exponential distribution. We used the parameter $\lambda = 1/1000$, therefore the expected value of the

size of the jobs were 1000, the variance was 1000000. We generated the penalties by exponential distribution as well, the used parameter was $\lambda = 1/100$, therefore the expected value of the penalty of the jobs were 100, the variance was 10000.

- Test D: In this case we generated the size of the jobs by hyperexponential distribution. We used two value $\lambda_1 = 1/800$, $p_1 = 1/2$, $\lambda_1 = 1/1200$, $p_2 = 1/2$. Therefore the expected value of the size of the jobs is $p_1/\lambda_1 + p_2/\lambda_2 = 1000$, the variance was 1080000. We generated the penalties by hyperexponential distribution as well, te used parameters were $\lambda_1 = 1/80$, $p_1 = 1/2$, $\lambda_1 = 1/120$, $p_2 = 1/2$, therefore the expected value of the penalty of the jobs was 100, the variance was 10800.

- Test E: In this case we generated the size of the jobs by Erlang-2 distribution. The used parameter was $\lambda = 500$, thus the expected size was 1000, the variance was 500000. We generated the penalties by Erlang-2 distribution as well, the used parameter was $\lambda = 1/50$, therefore the expected value of the penalty of the jobs was 100, the variance was 5000.

- Test F: In this case we generated the size of the jobs by bounded pareto distribution. We used the distribution $B(k, p, \alpha)$ where k (the bound on the smallest job size) was 1, p (the bound on the longest job size) was 20000, α is chosed to get the expected size 1000. We used the same distribution for penalties with upper bound 1000 and expected value 100.

In the case of the real data sets we used two subcases: the large input contained the full list of the jobs, in the small input we only used the first 20 percent of the jobs. We used 10 machines. The results are summarized in table 1. In the randomly generated tests we also investigated 2 subclasses. In the smaller size inputs 10000 jobs, in the larger size inputs one million jobs were generated and again we used 10 machines in the test. For every class we performed 100 randomly generated tests, the average results are summarized in table 2.

Table 1: The cost for real data

	A(Small)	A(Large)	B(Small)	B(Large)
Greedy	883549	4326745	2813479	13976483
RTP	843756	4117672	2889437	14245263
PAROLE	843679	4118935	2876559	14169854

We also investigated the number of rejected jobs. There was a big difference between test A and test B, in test A about 25 percent of the jobs was rejected and in test B around 42. This was expected since in test B the ratio of the expected values of the penalties and the jobs are smaller (in test A this ratio is 0,105 and in

Table 2: The average costs for randomly generated inputs

	C(Small)	C(Large)	D(Small)	D(Large)
Greedy	7487543	713456754	7964338	701341876
RTP	6822435	674523268	7657424	731735218
PAROLE	7014345	698764525	7776885	707823932
	E(Small)	E(Large)	F(Small)	F(Large)
Greedy	5698614	498238711	6718917	614492728
RTP	5254355	476521848	6396234	592718325
PAROLE	5317817	475632194	6512922	595598238

test B it is 0.037). In the randomly generated tests the ratio of the rejected jobs moved between 22 and 30 percent.

Considering the behavior of the algorithms we can observe that in each cases Greedy rejected the less jobs, and RTP rejected the most jobs. It is what we expected RTP and PAROLE reject each job which is rejected by Greedy.

If we consider the two type of costs then Greedy pays the less penalty and RTP the most. On the other hand only a small difference appeared in the amount of the paid penalty, it was less than 2 percent in all cases. In most tests the penalty is between 35 and 45 percent of the total cost, in the case of test B this ratio is larger, 60 percent.

4.2 Analysis of the results

Evaluating the results of the tests we can make the following observations:

- In most cases there is only small difference in the efficiency of the algorithms. The largest ratio of the costs occurred in test C(Small), where the ratio Greedy/RTP is 1.098. In most cases the ratio of the best and worst solution is below 1.05.

- *RTP* gave the best results in 6 test cases, PAROLE in 4 test cases and Greedy in 2 cases. On the other hand we note that PAROLE never resulted the worst result among the three algorithms. Thus we can conclude that PAROLE either gives the best result or its performance is between the performance of the other algorithms.

- The experimental results show that Greedy has better performance for the larger inputs. We think so that the reason of this property is that for large inputs Greedy can correct more easily the consequences of its bad decisions (when it accepts a large job, which actually should be rejected).

- We excepted that Greedy rejects the less jobs, and it has the less penalty cost and RTP pays the most penalty cost. Our tests confirmed this assumption.

It seems that PAROLE is more careful in rejecting jobs than RTP. On the other hand we can conclude that there are not big differences in the number of rejected jobs, in the test cases the three algorithms have similar behavior.

Summarizing the results of our experimental analysis we can conclude that the performance of PAROLE is usually between the performances of the other algorithms, therefore it can be useful in online computation where we have no information about the input.

5 Conclusions and further questions

In this paper we presented a new algorithm for the solution of the online scheduling with rejection problem. The efficiency of the algorithm is studied by an experimental analysis. The analysis shows that the algorithm gives good results on randomly generated inputs and on real data. Considering the algorithm defined in this paper some further open questions arise, we list them below.

The most important question is to characterize the complexity of finding the optimal value of the parameter of RTP. We conjecture so that this problem is NP-hard. It would be interesting to find better algorithms than CHOOSE to generate the next value of the parameter, we think so that such algorithm might improve significantly the efficiency of the algorithm. Finally we note that there exist further online problems where the best algorithm belongs to a class of algorithms and it is defined by fixing a parameter value which achieves the best competitive ratio. It is also an interesting question whether the idea of parameter learning is useful for such problems.

References

[1] Albers, S. and Schröder, B. An Experimental Study of Online Scheduling Algorithms. *ACM Journal of Experimental Algorithms*, article 3, 2002.

[2] Bartal, Y., Leonardi, S., Marchetti-Spaccamela, A., Sgall, J. and Stougie, L. Multiprocessor scheduling with rejection, *SIAM Journal on Discrete Mathematics*, 13:64–78, 2000.

[3] Bilo, V., Flammini, M. and Giovannelli, R. Experimental analysis of online algorithms for the bicriteria scheduling problem, *J. Parallel Distrib. Comput.*, 64:1086–1100, 2004.

[4] Borodin, A. and El-Yaniv, R. *Online Computation and Competitive Analysis*, Cambridge University Press, 1998.

[5] Correa, J.R., and Wagner, M. LP-Based Online Scheduling: From Single to Parallel Machines *Mathematical Programming*, to appear (prelinimary version in Proceedings of IPCO 2005 LNCS 3509, 186–209).

[6] Dósa, Gy. and He, Y. Scheduling with machine cost and rejection, *Journal of Combinatorial Optimization*, 12(4):337–350, 2006.

[7] Engels, D.V, Karger, D.R., Kolliopoulos, S.G., Sengupta, S., Uma, R.N. and Wein, J. Techniques for scheduling with rejection, *Journal of Algorithms*, 49:175–191, 2003.

[8] Epstein, L. and Sgall, L. Approximation schemes for scheduling on uniformly related and identical parallel machines, *Algorithmica*, 39(1):43–57, 2004.

[9] Fiat, A. and Woeginger, G.J., editors, *Online algorithms: The State of the Art, LNCS 1442*, Springer-Verlag Berlin, 1998

[10] Graham, R.L. Bounds for certain multiprocessor anomalies, *Bell System Technical Journal*, 45:1563–1581, 1966.

[11] Imreh, Cs. Scheduling problems on two sets of identical machines, *Computing*, 70:277–294, 2003.

[12] Imreh:, Cs. Competitive analysis, In Iványi, J., editor, *Algorithms of Informatics Volume 1*, pages 395–428, Budapest 2007, mondAt.

[13] Nagy-György, J. and Imreh, Cs. On-line scheduling with machine cost and rejection, *Discrete Applied Mathematics*, 155: 2546–2554, 2007.

[14] Seiden, S.S. Preemptive Multiprocessor Scheduling with Rejection, *Theoretical Computer Science*, 262:437–458, 2001.

[15] Sgall, J. On-line scheduling, In Fiat, A. and Woeginger, G.J., editors *Online algorithms: The State of the Art, LNCS 1442*, pages 196-231, Berlin, 1998, Springer Verlag.

Acta Cybernetica **19** (2009) 135–146.

Determining Initial Bound by "Ray-method" in Branch and Bound Procedure

Anett Rácz*

Abstract

In this paper we present an algorithm for determining initial bound for the Branch and Bound (B&B) method. The idea of this algorithm is based on the use of "ray" as introduced in the "ray-method" developed for solving integer linear programming problems [11], [12]. Instead of solving an integer programming problem we use the main idea of the ray-method to find an integer feasible solution of an integer linear programming problem along the ray as close to an optimal solution of the relaxation problem as possible. The objective value obtained in this manner may be used as an initial bound for the B&B method. It is well known that getting a "good bound" as soon as possible can often significantly increase the performance of the B&B method.

Keywords: integer programming, branch and bound, ray-method, initial bound

1 Introduction

It is well known that the performance of the B&B method mainly depends on the following three main factors:

- the rule used to choose the "branching" variable,
- the strategy used for generating binary search tree and
- the value of the initial bound.

Generally speaking, while the branching variable and strategy determine the size of the binary tree to be generated, getting a "good" bound as soon as possible can dramatically reduce the size of the tree to be considered, since the bound is used to prune those parts of the tree where the value of the objective function cannot be better than the bound.

Numerous efforts have been made in the past decades to investigate the general properties and behavior of the B&B method, e.g. [3, 6, 7, 13, 14, 16, 17, 18], to improve its computational efficiency, e.g. [4, 8, 9, 10, 15], to maximize its performance in different computational environments, see for example [5, 19], etc.[1]

*Department of Applied Mathematics and Probability Theory, Faculty of Informatics, University of Debrecen, 4032 Debrecen Egyetem tér 1., Hungary, E-mail: anett.racz@inf.unideb.hu

[1]The list of the relevant literature is so long, that in the framework of the current paper there is not enough space even to begin to cover all of the relevant items.

However, there is still some research being done in order to develop alternative algorithms for problems involving discrete variables. One of such investigations was performed in the Computer Center of the Russian Academy of Sciences [11, 12]. Schematically, the method developed by Khachaturov et al. is based on the search of "better" feasible integer solutions in some special set along some direction called a "ray". In this paper we present a new procedure for determining an initial bound for the branch and bound method which uses the basic ideas of the ray-method described in [11, 12].

The paper is organized as follows. In Section 2 we briefly overview the original ray-method – its mathematical background, its general scheme, different ways to define the ray, and implementation issues. In Section 3 we explain the main idea of the algorithm proposed: here we define the ray, then describe the main steps of the procedure, and finally, using a small illustrative numerical example, we show how this algorithm may be utilized. Section 4 summarizes my conclusions and my plans for future research.

2 The original *"Ray-method"*

Originally, the method was developed for a non-linear integer programming problem of the following form:

$$f(x) \longrightarrow \min \tag{1}$$

st.

$$x \in S \subset R^n, \tag{2}$$

$$x = (x_1, x_2, \ldots, x_n), \quad x_j - \text{ integer}, \ j = 1, 2, \ldots, n, \tag{3}$$

where feasible set

$$S = \{x \in R^n \mid g_i(x) \leq b_i, \ i = 1, 2, \ldots, m\}$$

is a convex, bounded and non-empty set, objective function $f(x)$ is non-linear differentiable $\forall x \in S$ and bounded from below on S, functions $g_i(x)$, $i = 1, 2, \ldots, m$, are non-linear and differentiable.

2.1 The mathematical background

Definition 1. *Let be given a feasible integer point $x^0 \in R^n$. We will say that integer point x' belongs to neighborhood set $O(x^0)$, i.e. $x' \in O(x^0)$ if values*

$$\left|x_1^0 - x_1'\right|, \quad \left|x_2^0 - x_2'\right|, \quad \ldots, \quad \left|x_n^0 - x_n'\right|$$

are relative primes.

Note that $O(x^0)$ has the following obvious properties.

Property 1. $x^0 \notin O(x^0)$.

Property 2. *If point $x' \in O(x^0)$, then on the straight line open segment (x^0, x') there is no integer point, i.e. there is no integer point x such that*

$$x = x^0 + \lambda(x' - x^0), \quad 0 < \lambda < 1 .$$

Let $S(x^0)$ denote the subset of feasible set S, where $f(x)$ is strictly less than $f(x^0)$, i.e.

$$S(x^0) = \{x \in S \mid f(x) < f(x^0)\}.$$

Using this notation we can formulate the following optimization criteria.

Theorem 1. *Feasible integer point x^0 is an optimal solution for problem (1)-(3) if and only if*

$$O(x^0) \cap S(x^0) = \emptyset \tag{4}$$

Proof: see [11], [12].

2.2 The general scheme

Let x^0 be an integer feasible solution for problem (1)-(3). In accordance with the main idea of the method we have to find such an integer point $x' \in O(x^0)$ that $x' \in O(x^0) \cap S(x^0)$. If $O(x^0) \cap S(x^0) = \emptyset$, then x^0 is an optimal solution for problem (1)-(3). The problem is solved.

Otherwise, i.e. if $O(x^0) \cap S(x^0) \neq \emptyset$, then we solve the following one-variable minimization problem:

$$f(\lambda) = f(x^0 + \lambda(x' - x^0)) \longrightarrow \min \tag{5}$$

$$x^0 + \lambda(x' - x^0) \in S \tag{6}$$

$$\lambda \geq 0. \tag{7}$$

Since both points x^0 and x' belong to the convex bounded non-empty feasible set S, constraint (6) defines the segment of straight line $x^0 + \lambda(x' - x^0)$, which belongs to S. So constraints (6) and (7) determine a non-empty bounded subset of S. Since original objective function $f(x)$ is continuous $\forall x \in S$ and bounded from below, it means that function $f(\lambda)$ is continuous and bounded too on any subset of S. The latter means that problem (5)-(7) is solvable and may be solved by any suitable numerical method. Let λ_{min} be its optimal solution and $[\lambda_{min}]$ denote its integer part. Concerning value $[\lambda_{min}] + 1$ it may occur that

$$x^0 + ([\lambda_{min}] + 1)(x' - x^0) \notin S, \tag{8}$$

or

$$f(x^0 + ([\lambda_{min}] + 1)(x' - x^0)) \geq f(x^0 + [\lambda_{min}](x' - x^0)) . \tag{9}$$

Now we construct the following point:

$$x'' = \begin{cases} x^0 + [\lambda_{min}](x' - x^0), & \text{if (8) or (9) takes place,} \\ x^0 + ([\lambda_{min}] + 1)(x' - x^0), & \text{otherwise.} \end{cases}$$

In other words, first, solving problem (5)-(7) we search the minimal value of function (5) on the ray beginning from point x^0 and passing through point x'. Then on this ray we have to find an integer feasible point x'' for which $f(x'')$ is most close to value $f(\lambda_{min})$. In the next step we denote point x'' by x^0 and repeat the process. The new set $S(x^0)$ differs from the previous one by only one constraint $f(x) <= f(x^0) = f(x'')$.

Since the number of integer points in feasible set S is bounded, the process will terminate in a finite number of iterations.

2.3 Different rules to define the ray

Let point x^0 be an integer feasible solution for problem (1)-(3), i.e. $x^0 \in S$. We will say that $L = \{x \in R^n \mid x = x^0 + \lambda l,\ \lambda \geq 0\}$, where $l = (l_1, l_2, \ldots, l_n) \in R^n$, is a ray, if there is $\lambda' > 0$ such that $f(x^0 + \lambda' l) < f(x^0)$. Using this notation, we describe here the following three ways by [12] for constructing a ray.

Procedure 1: Let x^* be the optimal solution of the relaxation problem (i.e. the problem without the integrality constraints) (1)-(2). Define ray L as

$$L = \{x \in R^n \mid x = x^0 + \lambda\,(x^* - x^0),\ \lambda \geq 0\}. \tag{10}$$

Procedure 2: Calculate the gradients $\nabla g_i(x)$ at the point x^0 for all $g_i(x)$, $i = 1, 2, \ldots, m$, functions, and introduce rays L_i, $i = 1, 2, \ldots, m$, in the following way:

$$L_i = x^0 + \lambda\, \nabla g_i(x^0), \quad \lambda \geq 0,$$

if exists such $\lambda' > 0$, that $f(x^0 + \lambda'\, \nabla g_i(x^0)) < f(x^0)$. And

$$L_i = x^0 - \lambda\, \nabla g_i(x^0), \quad \lambda \geq 0,$$

otherwise.

Procedure 3: Choose the following formula for the ray.

$$L = x^0 - \lambda\, \nabla f(x^0), \quad \lambda \geq 0. \tag{11}$$

2.4 Implementation issues

In contrast to the transparency of the theoretical background for the method there are serious difficulties with its implementation and computational efficiency. The main and most serious of them is checking optimality criteria for a given feasible integer point x^0 since set $O(x^0)$ may contain a huge number of integer points. Note that, as was mentioned above, these integer points were selected on the basis of the usage of relative prime numbers, so determining these integer points may be a very hard and computationally very expensive problem. This is why the developers of the method for numerical experiments used different approximate variants of the method and tested it using problems of relatively small size (up to 100 constraints $\times$ 120 variables).

3 The new method proposed

As was mentioned above, the ray-method was originally developed as a method for solving non-linear integer programming problems. Using the main ideas of the method, below we propose a new algorithm which may be used for determining the initial bound in the branch and bound method when solving integer linear programming problems.

3.1 Preliminaries

Consider the following pure integer linear programming minimization problem:

$$f(x) = \sum_{j=1}^{n} c_j x_j \longrightarrow \min \tag{12}$$

st.

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i, \quad i = 1, 2, \ldots, m, \tag{13}$$

$$x_j \geq 0, \quad \text{integer}, \quad j = 1, 2, \ldots, n. \tag{14}$$

Here and in what follows we assume that relaxation problem (12)-(14) is solvable (i.e. has a non-empty feasible set and objective function $f(x)$ over the feasible set has a finite lower bound) and vector

$$x^{min} = (x_1^{min}, x_2^{min}, \ldots, x_n^{min})$$

is its relaxation non-integer solution. Furthermore, we suppose that the corresponding maximization relaxation problem is solvable too, and

$$x^{max} = (x_1^{max}, x_2^{max}, \ldots, x_n^{max})$$

is its optimal solution. These two assumptions play a very important role in our algorithm since we use points x^{min} and x^{max} to determine the ray for indicating the direction of the search. Moreover we assume that $x^{min} \neq x^{max}$.

3.2 Main steps

Using the given notation, the algorithm proposed may be described in the following steps.

0. Initial point: Let us denote point x^{min} by x^0, and $l = (l_1, l_2, \ldots, l_n)$, where $l_j = x_j^{max} - x_j^{min}$, $j = 1 \ldots n$.

1. Ray: Define the ray in the following way:

$$L = x^0 + \lambda(x^{max} - x^0), \quad 0 \leq \lambda \leq 1 .$$

Note that since feasible set S is convex, it means that all points of L are elements of set S.

2. Constructing set $O(x^0)$: Let J^0 be a set of indexes of integer components of x^0, i.e. $J^0 = \{j \in J |\ x_j^0 = [x_j^0]\}$, where $J = \{1, 2, \ldots, n\}$. We define set $O(x^0)$ as the set of such points x which satisfy the following constraints:

$$\left.\begin{array}{lcll} [x_j^0] & \leq x_j \leq & [x_j^0]+1, & \text{if } j \notin J^0\,, \\ [x_j^0] & \leq x_j \leq & [x_j^0]+1, & \text{if } j \in J^0 \text{ and } l_j > 0\,, \\ [x_j^0]-1 & \leq x_j \leq & [x_j^0], & \text{if } j \in J^0 \text{ and } l_j < 0\,, \\ & x_j = & [x_j^0], & \text{if } j \in J^0 \text{ and } l_j = 0\,. \end{array}\right\} \tag{15}$$

Generally speaking $O(x^0)$ is the unit-cube containing the point x^0. If $J^0 = \emptyset$, then the dimension of this unit-cube is n.

Before starting the iterations let us define the point $x' := x^0$ and calculate the first perforation point $P_{act} = (p_1, p_2, \ldots, p_n)$ solving the following optimization problem:

$$\lambda \longrightarrow \max \tag{16}$$

st.

$$p_j = x_j^0 + \lambda l_j \quad j = 1, 2, \ldots, n, \tag{17}$$

$$\left.\begin{array}{lcll} [x_j^0] & \leq p_j \leq & [x_j^0]+1, & j \notin J^0\,, \\ [x_j^0] & \leq p_j \leq & [x_j^0]+1, & j \in J^0 \text{ and } l_j > 0\,, \\ [x_j^0]-1 & \leq p_j \leq & [x_j^0], & j \in J^0 \text{ and } l_j < 0\,, \\ & p_j = & [x_j^0], & j \in J^0 \text{ and } l_j = 0\,. \end{array}\right\} \tag{18}$$

3. Shifting: Now we enter new variables $y_j = x_j - [x_j']$, $j = 1, 2, \ldots, n$, and construct new feasible set S' in the following way

$$S': \quad \sum_{j=1}^{n} a_{ij} y_j \leq b_i', \quad i = 1, 2, \ldots, m\,,$$

where

$$b_i' = b_i - \sum_{j=1}^{n} a_{ij} [x_j'], \quad i = 1, 2, \ldots, m\,.$$

Obviously, set S' is the intersection of the current set $O(x')$ and feasible set S shifted to point 0. Then we solve the following 0-1 LP problem

$$f'(y) = \sum_{j=1}^{n} c_j y_j + c_0 \rightarrow \max \tag{19}$$

st.

$$\sum_{j=1}^{n} a_{ij} y_j \leq b_i', \quad i = 1, 2, \ldots, m, \tag{20}$$

$$y_j = 0/1, \quad j = 1, 2, \ldots, n, \tag{21}$$

where

$$c_0 = \sum_{j=1}^{n} c_j [x'_j] \,,$$

using **Balas'** additive algorithm (implicit enumeration) [2]. If problem (19)-(21) has 0-1 optimal solution y^*, then we determine vector $x^* = (x_1^*, x_2^*, \ldots, x_n^*)$, where

$$x_j^* = y_j^* + [x'_j], \quad j = 1, 2, \ldots, n \,,$$

and use value $f(x^*) = f'(y^*)$ as an initial bound for the branch and bound method. **Stop.**
Otherwise,

4. **Perforation point:** We determine point P_{next} where the ray "perforates" the hull of the next unit-cube along the ray solving the following optimization problem:

$$\lambda \longrightarrow \max \tag{22}$$

st.

$$x_j = x_j^0 + \lambda l_j \quad j = 1, 2, \ldots, n \,, \tag{23}$$

$$\left.\begin{array}{lcll} [p_j] & \leq x_j \leq & [p_j] + 1, & j \notin J' \,, \\ [p_j] & \leq x_j \leq & [p_j] + 1, & j \in J' \text{ and } l_j > 0 \,, \\ [p_j] - 1 & \leq x_j \leq & [p_j], & j \in J' \text{ and } l_j < 0 \,, \\ & x_j = & [p_j], & j \in J' \text{ and } l_j = 0 \,, \end{array}\right\} \tag{24}$$

where $J' = \{j \in J |\, p_j = [p_j]\}$. Here constraints (23) and (24) provide $P_{next} \in L$ and $P_{next} \in O(P_{act})$, correspondingly. Obviously, this problem is solvable, i.e. has a non-empty feasible set and its objective function is bounded from above. Let $P_{next} = (p'_1, p'_2, \ldots, p'_n)$ solve problem (22)-(24) and λ' be the maximal value of objective function (22).

Let us define middle point x' of the section $[P_{act};\ P_{next}]$:

$$x'_j = \frac{p_j + p'_j}{2} \quad j = 1, 2, \ldots, n \,. \tag{25}$$

Furthermore, we do not need point P_{act} any more, so we overwrite it with the value of P_{next}, i.e. $P_{act} := P_{next}$.

5. **Next unit-cube:** Having point x' we can determine the next unit-cube along the ray using the following rule:

$$\left.\begin{array}{lcll} [x'_j] & \leq x_j \leq & [x'_j] + 1, & \text{if } j \notin J' \,, \\ [x'_j] & \leq x_j \leq & [x'_j] + 1, & \text{if } j \in J' \text{ and } l_j > 0 \,, \\ [x'_j] - 1 & \leq x_j \leq & [x'_j], & \text{if } j \in J' \text{ and } l_j < 0 \,, \\ & x_j = & [x'_j], & \text{if } j \in J' \text{ and } l_j = 0 \,, \end{array}\right\} \tag{26}$$

where $J' = \{j \in J |\ x'_j = [x'_j]\}$. Go to step 3.

Since the number of unit-cubes "perforated" by the ray is finite, the process will terminate in a finite number of iterations. It may occur that when determining next perforation point x' we obtain $\lambda' > 1$. It means that unit-cubes constructed along the ray do not contain any integer feasible point for original problem (12)-(14). It means the method fails.

3.3 An illustrative numerical example

Here the main steps of the method proposed using a small numerical example are illustrated. The method proposed was partially implemented in the frame of the educational linear and linear-fractional package WinGULF [1]. The package has numerous options for the B&B method - we can choose the direction of the search (first left node and then right one or vice versa), different rules for selecting a branching variable (for example, "fractional part most close to 0.5", "smallest fractional part", "biggest fractional part", "smallest value", "biggest value", etc.), user defined initial bound, etc. When testing the method proposed, most of the options built in were used.

Consider the following numerical example:

$$
\begin{array}{lrrrcr}
f(x) = & 20x_1 + & 21x_2 + & 18x_3 & \longrightarrow & \min \\
\text{st.} & & & & & \\
& 9x_1 + & 1.5x_2 + & 7x_3 & \leq & 1350\,, \\
& 5.5x_1 + & 1x_2 + & 9x_3 & \geq & 1250\,, \\
& -4.5x_1 - & 10x_2 + & 2.5x_3 & \leq & -1050\,, \\
& & x_1,\ x_2,\ x_3 - & \text{integer.} & &
\end{array}
$$

Solving both (minimization and maximization) relaxation problems we obtain the following:

$$
\begin{array}{lcrrr}
x^{min} & = & (\ 65.042, & 97.799, & 88.274)\,, \\
x^{max} & = & (\ 0.000, & 523.076, & 80.769)\,, \\
L & = & (-65.042, & 425.277, & -7.504)\,.
\end{array}
$$

Let us denote x^{min} with x^0 and construct set $O(x^0)$, i.e. the following unit-cube:

$$
O(x^0): \begin{cases} 65 \leq x_1 \leq 66\,, \\ 97 \leq x_2 \leq 98\,, \\ 88 \leq x_3 \leq 89\,. \end{cases}
$$

So we can construct the first shifted problem:

$$
\begin{array}{lrrrrrrl}
f'(y) & = & 20y_1+ & 21y_2+ & 18y_3 & + & 4921 & \longrightarrow \max \\
\text{st.} & & & & & & & \\
 & & 9y_1+ & 1.5y_2+ & 7y_3 & \leq & 3.5\,, & \\
 & & 5.5y_1+ & 1y_2+ & 9y_3 & \geq & 3.5\,, & \\
 & & -4.5y_1- & 10y_2+ & 2.5y_3 & \leq & -7.5\,, & \\
 & & & y_1,\ y_2,\ y_3 & - & 0/1 & &
\end{array}
$$

and try to solve it. Since the problem is infeasible, we have to determine the next unit-cube along the ray. In order to obtain the next unit-cube first we solve the following problem (see (23)-(24)):

$$\lambda \rightarrow \max$$

st.

$$
\left.
\begin{aligned}
x_1 &= 65.042 + \lambda(-65.042)\,, \\
x_2 &= 97.799 + \lambda(425.277)\,, \\
x_3 &= 88.274 + \lambda(-7.504)\,,
\end{aligned}
\right\}
$$

$$
\left.
\begin{aligned}
65 \leq\ & x_1\,, \\
& x_2 \leq 98\,, \\
88 \leq\ & x_3
\end{aligned}
\right\}
$$

and obtain the first perforation point P_1:

$$\lambda' = 0.00047, \quad P_1 = (65.011,\ 98,\ 88.270)\,.$$

To find the next perforation point P_2 we have to solve the following problem:

$$\lambda \rightarrow \max$$

st.

$$
\left.
\begin{aligned}
x_1 &= 65.042 + \lambda(-65.042)\,, \\
x_2 &= 97.799 + \lambda(425.277)\,, \\
x_3 &= 88.274 + \lambda(-7.504)\,,
\end{aligned}
\right\}
$$

$$
\left.
\begin{aligned}
65 \leq\ & x_1\,, \\
& x_2 \leq 99\,, \\
88 \leq\ & x_3\,,
\end{aligned}
\right\}
$$

so we obtain

$$\lambda'' = 0.00064, \quad P_2 = (65,\ 98.07,\ 88.26)\,.$$

Using these points P_1 and P_2 we find the middle point

$$x' = (65.006,\ 98.03,\ 88.26)\ .$$

This point allows us to construct the next shifted problem:

$$\begin{array}{rcrrrcrl} f'(y) & = & 20y_1+ & 21y_2+ & 18y_3 & + & 4942 & \longrightarrow \max \\ \text{st.} & & & & & & & \\ & & 9y_1+ & 1.5y_2+ & 7y_3 & \leq & 2\ , & \\ & & 5.5y_1+ & 1y_2+ & 9y_3 & \geq & 2.5\ , & \\ & & -4.5y_1- & 10y_2+ & 2.5y_3 & \leq & -2.5\ , & \end{array}$$

$$y_1,\ y_2,\ y_3\ -\ 0/1\ .$$

This problem has no feasible solution.

Proceeding to the next perforation point P_3, we obtain the following optimization problem to solve

$$\lambda \rightarrow \max$$

st.

$$\left.\begin{array}{l} x_1 = 65.042 + \lambda(-65.042)\ , \\ x_2 = 97.799 + \lambda(425.277)\ , \\ x_3 = 88.274 + \lambda(-7.504)\ , \end{array}\right\}$$

$$\left.\begin{array}{rcl} 64 \leq & x_1\ , & \\ & x_2 & \leq 99\ , \\ 88 \leq & x_3\ . & \end{array}\right\}$$

We obtain $\lambda = 0.0028$ and the next perforation point $P_3 = (64.85,\ 99,\ 88.25)$. Therefore the next middle point is $x' = (64.93,\ 98.53,\ 88.26)$ and the shifted problem is as follows:

$$\begin{array}{rcrrrcrl} f'(y) & = & 20y_1+ & 21y_2+ & 18y_3 & + & 4922 & \longrightarrow \max \\ \text{st.} & & & & & & & \\ & & 9y_1+ & 1.5y_2+ & 7y_3 & \leq & 11\ , & \\ & & 5.5y_1+ & 1y_2+ & 9y_3 & \geq & 8\ , & \\ & & -4.5y_1- & 10y_2+ & 2.5y_3 & \leq & -2\ , & \end{array}$$

$$y_1,\ y_2,\ y_3\ -\ 0/1\ .$$

The optimal solution of this problem is $y^* = (0,\ 1,\ 1)$ and $f'(y^*) = 4961$. This value may be used as an initial bound. The corresponding integer point is $x^* = (64,\ 99,\ 89)$.

Note that the initial bound obtained (4961) is very close to the optimal value (after solving the problem we obtain 4959). Below is presented a table with results

obtained from WinGULF after running B&B method on this numerical example using different strategies (from left to right and vice versa) and different branching rules. *"Wo.I.B."* means without an initial bound and *"W.I.B."* means with an initial bound.

Branching variable	Left → Right		Right → Left	
	Wo.I.B.	W.I.B.	Wo.I.B.	W.I.B.
Minimal index	291	37	37	33
Maximal index	31	23	243	23
Max. value	33	25	245	25
Min. value	31	25	25	25
Max. fract. part	105	41	37	37
Min. fract. part	31	23	23	23
Most close to 0.5	31	25	25	25

4 Further work and ideas to improve efficiency

Some simple manual tests were performed. These preliminary tests show that it could be worth making more efforts in the topic and in performing computerized numerical tests with different LP problems including MIPLIB and other test collections. It is clear that test results depend very hard on the search strategy and branching rules used when running B&B. This is why it would be worth developing a special software application, which could be used when comparing the efficiency of the ray-method in different cases. Moreover, we would like to improve the ray-method too. One of the possible directions for further research may be an investigation connected with extending the main ideas of the method to the case of mixed integer programming problems. Another open question is ray-selection in the case when the maximization problem is unbounded.

References

[1] Bajalinov, E., *"Linear-fractional programming: theory, methods, software and applications"*, Kluwer, 2003.

[2] Balas, E., *"An additive algorithm for solving linear programs with zero-one variables"*, Operations Research, vol. 13, pp. 517-46, 1965.

[3] Berliner, H., *"The B tree search algorithm: A best-first proof procedure"*, Artificial Intell., vol. 12, no. 1, pp. 23-40, 1979.

[4] Borchers, B., Mitchell, J.E., *"An improved branch and bound algorithm for mixed integer nonlinear programs"*, Computers and Operations Research, vol. 21, issue 4, pp. 359-367, 1994.

[5] Gendron, B., Crainic, T.G., *"Parallel Branch-and-Bound Algorithms: Survey and Synthesis."*, Opererations Research, vol. 42 issue 6, pp. 1042-1066, 1994.

[6] Gupta, O. K., Ravindran, V., *"Branch and Bound Experiments in Convex Nonlinear Integer Programming"*, Management Science, vol. 31, no. 12, 1533-1546, 1985.

[7] Hawkins, D. M., *"Branch-and-Bound method"*, Encyclopedia of Statistical Sciences, John Wiley and Sons, 2006.

[8] Ibaraki, T., *"Computational efficiency of approximate branch-and-bound algorithms"*, Math. Oper. Res., vol. 1, no. 3, pp. 287-298, 1976.

[9] Ibaraki, T., *"Theoretical comparisons of search strategies in branch-and-bound algorithms"*, Int. J. Computer and Information Sciences, vol. 5, no. 4, pp. 315-343, 1976.

[10] Ibaraki, T., *"The Power of Dominance Relations in Branch-and-Bound Algorithms"*, Journal of the ACM (JACM), vol. 24, issue 2, pp. 264 - 279, 1977.

[11] Khachaturov, V.R., Mirzoyan, N.A. *"Solving problems of integer programming with ray-method."* Notes on applied mathematics, Computer Center of Soviet Academy of Science, 1987.

[12] Khachaturov, V.R., *"Combinatorial methods and algorithms for solving large-scale discrete optimization problems"*, Moscow, Nauka, 2000.

[13] Kumar, V., Kanal, L. N., *"A general branch and bound formulation for understanding and synthesizing and/or tree search procedures"*, Artificial Intell., vol. 21, no. 1-2, pp. 179-198, 1983.

[14] Lawler, E. L., Wood, D. E., *"Branch-And-Bound Methods: A Survey"*, Operations Research, vol. 14, no. 4, pp. 699-719, 1966.

[15] Linderoth, T. Savelsbergh, M. W. P., *"A computational study of branch and bound search strategies for mixed integer programming"*, INFORMS J. Computing, vol. 11, no. 2, 173-187, 1999.

[16] Mitten, L., *"Branch and bound methods: General formulation and properties"*, Operation Research, vol. 18, pp. 24-34, 1970.

[17] Smith, D.R., *"Random trees and the analysis of branch and bound procedures"*, Journal of the Association for computing machinery, vol.31, no.1, pp.163-188, 1984.

[18] Yu, C.-F., Wah, B.W., *"Stochastic modeling of branch-and-bound algorithms with best-first search"*, IEEE Transactions on Software Engineering, vol. SE-11, no. 9, pp. 922-934, 1985.

[19] Yu, C.-F., Wah, B.W., *"Efficient Branch-and-Bound Algorithms on a Two-Level Memory System"*, IEEE Transactions on Software Engineering, vol. 14, no. 9, pp. 1342-1356, 1988.

Regular Papers

Acta Cybernetica **19** (2009) 149–158.

A Fixed Point Theorem for Stronger Association Rules and Its Computational Aspects*

Gábor Czédli†

Abstract

Each relation induces a new closure operator, which is (in the sense of data mining) stronger than or equal to the Galois one. The goal is to give some evidence that the new closure operator is often properly stronger than the Galois one. An easy characterization of the new closure operator as a largest fixed point of an appropriate contraction map leads to a (modest) computer program. Finally, various experimental results obtained by this program give the desired evidence.

Keywords: Association rule, database, data mining, lattice, poset, context, concept lattice, formal concept analysis, Galois connection, functional dependency.

1 Introduction to a new closure operator

Although the terminology of formal concept analysis is frequently used in the first two sections, the paper belongs neither to formal concept analysis nor to other applied fields about data bases. These fields are used only as a part of motivations of the present study. However, in the converse direction, there is some hope that this section and mainly the next one may give some motivation to these fields. This section is devoted only to definitions and some basic properties; our motivations will be given in the next section.

Following Wille's terminology, cf. [11] or [7], a triplet

$$(A^{(0)}, A^{(1)}, \rho)$$

is called a *context* if $A^{(0)}$ and $A^{(1)}$ are nonempty sets and $\rho \subseteq A^{(0)} \times A^{(1)}$ is a binary relation. From what follows, we fix a context $(A^{(0)}, A^{(1)}, \rho)$ and let

$$\rho_0 = \rho \text{ and } \rho_1 = \rho^{-1}.$$

*This research was partially supported by the NFSR of Hungary (OTKA), grant no. T 049433 and K 60148

†University of Szeged, www.math.u-szeged.hu/∼czedli/. E-mail: czedli@math.u-szeged.hu

From now on, unless otherwise stated, i will be an arbitrary element of $\{0,1\}$. So whatever we say including i without specification, it will be understood as prefixed by $\forall i$. The set of all subsets of $A^{(i)}$ will be denoted by $P(A^{(i)})$.

It is often, especially in the finite case, convenient to depict our context in the usual form: a binary table with row labels from $A^{(0)}$, column labels from $A^{(1)}$, and a cross in the intersection of the x-th row and the y-th column iff $(x,y) \in \rho$. We will refer to this table as the *context table*. For example, a context is given by Table 1. (In an appropriate sense, cf. Table 2, this is the smallest interesting example.) The concrete meaning of this context is not relevant for this paper[1].

	b_1	b_2	b_3	b_4
a_1	×	×		
a_2	×		×	
a_3	×	×		×
a_4	×			
a_5		×	×	×

Table 1

A mapping $\mathcal{D}^{(i)} : P(A^{(i)}) \to P(A^{(i)})$ is called a *closure operator* if it is *extensive* (i.e., $X \subseteq \mathcal{D}^{(i)}(X)$ for all $X \in P(A^{(i)})$), *monotone* (i.e., $X \subseteq Y$ implies $\mathcal{D}^{(i)}(X) \subseteq \mathcal{D}^{(i)}(Y)$), and *idempotent* (i.e., $\mathcal{D}^{(i)}(\mathcal{D}^{(i)}(X)) = \mathcal{D}^{(i)}(X)$ for all $X \in P(A^{(i)})$). By a *pair of extensive operators* we mean a pair $\mathcal{D} = (\mathcal{D}^{(0)}, \mathcal{D}^{(1)})$ where $\mathcal{D}^{(i)} : P(A^{(i)}) \to P(A^{(i)})$ is an extensive mapping for $i = 0, 1$. If these mappings are closure operators then $\mathcal{D}$ is called a *pair of closure operators*.

If $\mathcal{D} = (\mathcal{D}^{(0)}, \mathcal{D}^{(1)})$ and $\mathcal{E} = (\mathcal{E}^{(0)}, \mathcal{E}^{(1)})$ are pairs of extensive operators then $\mathcal{D} \leq \mathcal{E}$ means that $\mathcal{D}^{(i)}(X) \subseteq \mathcal{E}^{(i)}(X)$ for all $i \in \{0,1\}$ and all $X \in P(A^{(i)})$.

Now, associated with $(A^{(0)}, A^{(1)}, \rho)$, we define some pairs of closure operators. The motivation will be given afterwards. For $X \in P(A^{(i)})$ let

$$X\rho_i = \{y \in A^{(1-i)} : \text{ for all } x \in X,\ (x,y) \in \rho_i\},$$

and, again for $X \in P(A^{(i)})$, define

$$\mathcal{G}^{(i)}(X) := (X\rho_i)\rho_{1-i} = \bigcap_{y \in X\rho_i} (\{y\}\rho_{1-i}) .$$

Then $\mathcal{G} = (\mathcal{G}^{(0)}, \mathcal{G}^{(1)})$ is the well-known *pair of Galois closure operators*, which plays the main role in formal concept analysis, cf. Wille [11] and Ganter and Wille [7]. The visual meaning of

$$\mathcal{G} = \mathcal{G}(A^{(0)}, A^{(1)}, \rho)$$

is the following. The maximal subsets of ρ of the form $U^{(0)} \times U^{(1)}$ with $U^{(i)} \subseteq A^{(i)}$ are called the (formal) *concepts*, cf. [11] or [7]. Pictorially, they are the maximal full rectangles $U^{(0)} \times U^{(1)}$ of the context table. (Full means that each entry is a

[1] This is a context about juggling, the details are at Publications/[77] in the author's web site.

cross.) For $X_i \in P(A^{(i)})$ take all maximal full rectangles $U^{(0)} \times U^{(1)}$ such that $X \subseteq U^{(i)}$, then $\mathcal{G}^{(i)}(X)$ is the intersection of all the $U^{(i)}$'s.

Now we define a sequence $\mathcal{C}_i$, $i = 0, 1, 2, \ldots$, of pairs of of closure operators. For $X \in P(A^{(i)})$ let

$$X\psi_i := \{Y \in P(A^{(1-i)}) : \text{ there is a surjection } \varphi : X \to Y \text{ with } \varphi \subseteq \rho_i\}.$$

Pictorially, the elements of $X\psi_i$ are easy to imagine. For example, let $i = 0$, i.e., let $X \subseteq A^{(0)}$ be a set of rows. Select a cross in each row of X, then the collection of the columns of the selected crosses is an element of $X\psi_0$, and each element of $X\psi_0$ is obtained this way. For example, if $X = \{a_1, a_2\}$ in Table 1 then $X\psi_0$ consists of $\{b_1\}$, $\{b_1, b_2\}$, $\{b_1, b_3\}$ and $\{b_2, b_3\}$.

Let $\mathcal{C}_0 = \mathcal{G}$. If $\mathcal{C}_n$ is already defined then let

$$\mathcal{C}^{(i)}_{n+1}(X) := \mathcal{C}^{(i)}_n(X) \cap \bigcap_{Y \in X\psi_i} \bigcup_{y \in \mathcal{C}^{(1-i)}_n(Y)} \{y\}\rho_{1-i}\,. \tag{1}$$

This defines the pair $\mathcal{C}_{n+1} = (\mathcal{C}^{(0)}_{n+1}, \mathcal{C}^{(1)}_{n+1})$.

The easiest way to digest formula (1) is to think of it pictorially. For example, let $i = 0$ and $X \subseteq A^{(0)}$, and suppose that $\mathcal{C}_n = (\mathcal{C}^{(0)}_n, \mathcal{C}^{(1)}_n)$ is already well-understood. Then a row z belongs to $\mathcal{C}^{(0)}_{n+1}(X)$ if and only if $z \in \mathcal{C}^{(0)}_n(X)$ and, in addition, for each set $Y \in X\psi_0$ of columns there is a column y in $\mathcal{C}^{(1)}_n(Y)$ such that y intersects the row z at a cross. (Notice that $X\psi_0$ has already been explained pictorially, $\mathcal{C}^{(0)}_n(X)$ and $\mathcal{C}^{(1)}_n(Y)$ are already well-known by assumption, and y need not be unique and it depends on Y.)

Finally, let

$$\mathcal{C} = (\mathcal{C}^{(0)}, \mathcal{C}^{(1)}) := (\bigwedge_{n=0}^{\infty} \mathcal{C}^{(0)}_n, \bigwedge_{n=0}^{\infty} \mathcal{C}^{(1)}_n),$$

which means that, for all $X \in P(A^{(i)})$,

$$\mathcal{C}^{(i)}(X) = \bigcap_{n=0}^{\infty} \mathcal{C}^{(i)}_n(X).$$

Although the above definitions look neither friendly nor natural at the first sight, they had a proper application in [3]; proper means that $\mathcal{C}$ was heavily used when proving a theorem which has nothing to do with the notion of $\mathcal{C}$. Notice also that it was routine to prove in [3] that we have indeed defined pairs of closure operators.

Lemma 1. *(cf. [3])* $\mathcal{C} = \mathcal{C}(A^{(0)}, A^{(1)}, \rho)$ *and* $\mathcal{C}_n = \mathcal{C}_n(A^{(0)}, A^{(1)}, \rho)$ $(n = 0, 1, \ldots)$ *are pairs of closure operators. Further,* $\mathcal{G} = \mathcal{C}_0 \geq \mathcal{C}_1 \geq \mathcal{C}_2 \geq \cdots \geq \mathcal{C}$.

It is well-known and goes back to Galois that, for each context $(A^{(0)}, A^{(1)}, \rho)$, the complete lattices $(\{X \in P(A^{(0)}) : \mathcal{G}^{(0)}(X) = X\}, \subseteq)$ and $(\{X \in P(A^{(1)}) :$

$\mathcal{G}^{(1)}(X) = X\}, \subseteq)$ are dually isomorphic. The analogous statement is far from being true for C; indeed, in case of the context given by Table 1, $|\{X \in P(A^{(0)}) : C^{(0)}(X) = X\}| = 12$ while $|\{X \in P(A^{(1)}) : C^{(1)}(X) = X\}| = 10$.

From now on *we always assume* that $(A^{(0)}, A^{(1)}, \rho)$ is *finite*. Then there are only finitely many pairs of operators, whence there is a smallest n with $C = C_n = C_{n+1} = C_{n+2} = \cdots$. This raises the natural question how large this n can be. To shed more light on C, we answer this question below.

Let us say that $(A^{(0)}, A^{(1)}, \rho)$ is a decomposable context if there are nonempty sets $B^{(i)}$ and $C^{(i)}$ with $B^{(i)} \cup C^{(i)} = A^{(i)}$ and $B^{(i)} \cap C^{(i)} = \emptyset$ such that

$$\rho = (\rho \cap (B^{(0)} \times B^{(1)})) \cup (\rho \cap (C^{(0)} \times C^{(1)})).$$

Otherwise $(A^{(0)}, A^{(1)}, \rho)$ is called an *indecomposable context*. We say that it is a *uniform context* if $|\{x\}\rho_i| = |\{y\}\rho_i|$ for all $x, y \in A^{(i)}$. For example, all finite block designs (P, B, I) and, in particular, all finite projective spaces (P, L, I) are uniform contexts. If $|\{x\}\rho_i| = |\{y\}\rho_i| = 2$ for all $x, y \in A^{(i)}$ then we speak of a *2-uniform context* In the terminology of context tables, a context is 2-uniform iff there are exactly two crosses in each row and in each column.

Now, as an anonymous referee of [4] suggested, it is routine to extract from [3] that even if we restrict ourselves to finite indecomposable 2-uniform contexts, arbitrarily large numbers n with $\mathcal{G} = C_0 > C_1 > C_2 > C_3 \cdots C_n$ occur. We close this section by recalling an open problem about C; for motivation and a possible application cf. [3].

Problem 1. *Is it true that for each indecomposable uniform context* $(A^{(0)}, A^{(1)}, \rho)$ *with* $|A^{(0)}| \geq 3$ *and* $|A^{(1)}| \geq 3$ *there exists an* $i \in \{0, 1\}$ *and there are* $x, y, z \in A^{(i)}$ *such that*

$$C^{(i)}(\{x, y\}) \cap C^{(i)}(\{y, z\}) \cap C^{(i)}(\{z, x\}) = \emptyset?$$

2 Motivations

Now we give our motivations, and this will explain why "association rule" occurs in the title of the paper. Closure operators have been playing an important role in the theory of relational databases and knowledge systems for a long time, cf. e.g., Caspard and Monjardet [2] for a survey. Nowadays most investigations of this kind belong to formal concept analysis, cf. Ganter and Wille [7] for an extensive survey. The theory of mining association rules goes back to Agrawal, Imielinski and Swami [1]; Lakhal and Stumme [8] gives a good account on the present status of this field.

For a data miner, the context is a *huge* binary database, and mining association rules is a popular knowledge discovery technique for warehouse basket analysis. In this case $A^{(0)}$ is the set of costumers' baskets, $A^{(1)}$ is the set of items sold in the warehouse, and the task is to figure out which items are frequently bought together. This information, expressed by so-called "association rules", can help the warehouse in developing appropriate marketing strategies. For example,

$$\{\text{cereal, coffee}\} \rightarrow \{\text{milk}\}$$

is an association rule (in many real warehouses), and this association rule says that, with a given probability p, costumers buying cereal and coffee also buy milk. When milk $\in \mathcal{G}^{(1)}\{\text{cereal, coffee}\}$ then this probability is 1 and we speak about a *strong* association rule.

However, the importance of looking for the hidden regularities and rules is not restricted only to *huge* databases. The success of formal concept analysis, cf. Ganter and Wille [7], or Mendeleyev's classical periodic system of chemical elements show that exploring some rules in *small* databases may also lead to important results. From this aspect, the present paper offers $\mathcal{C}$, a mathematical tool, to formulate some regularities in abstract contexts. Since $\mathcal{C} \leq \mathcal{G}$, the "association rules" corresponding to $\mathcal{C}$ are *stronger* than the previously mentioned ones. It would be nice to find some concrete contexts outside mathematics, say in natural or social sciences, where $\mathcal{C}$ has some real applications and gives new insights that $\mathcal{G}$ does not. This is much beyond the scope of the present paper but there is a hope, for finding associations is an integral part of any creative activity.

Now we use the context given by Table 1 to develop our ideas further. Let $X = \{a_1, a_2\} \subseteq A^{(0)} = \{a_1, \ldots, a_5\}$. Then $\{a_1, \ldots, a_4\} \times \{b_1\}$ is the only relevant maximal full rectangle to compute $\mathcal{G}^{(0)}(X) = \{a_1, \ldots, a_4\}$. Since $Y = \{b_2, b_3\} \in X\psi_0$ but there is no $y \in \mathcal{G}^{(1)}(Y) = \{b_2, b_3, b_4\}$ with $a_4 \in \{y\}\rho_1$, formula (1) gives $a_4 \notin C_1^{(0)}(X)$. After the trivial and therefore omitted details we can easily see that $\mathcal{C} = \mathcal{C}_1$ and $\mathcal{C}^{(0)}(X) = \{a_1, a_2, a_3\}$.

Suppose our whole knowledge is decoded in the context and we have to associate an element with X. Usually we want an element outside X, and we look for something similar, i.e., we want an element which shares the common attributes of the elements of X. So the first answer is that we should associate some element of $\mathcal{G}^{(0)}(X) \setminus X = \{a_3, a_4\}$. This way we obtain more than one element, but we may want to chose only a single one. For example, which of a_3 and a_4 should a scientist choose if the context represents something in his research field and choosing both is not permitted[2]? The unique element a_3 of $\mathcal{C}^{(0)}(X) \setminus X$? The other element, a_4? We cannot answer to this question of *decision making* in full generality.

The main motivation to investigate $\mathcal{C}$ is that [3], where $\mathcal{C}$ has a proper application, witnesses that $\mathcal{C}$ is useful in algebra.

We have seen that there is chance that $\mathcal{C}$ gives some insights and tools that $\mathcal{G}$, successfully used various fields, does not. But these are just hopes at present, and as a very first step to justify these expectations the paper gives some partial answers to the question how often $\mathcal{C}$ is different from $\mathcal{G}$.

[2]For the mentioned concrete meaning of Table 1 the beginner may ask which one of a_3 and a_4 is easier to learn after a_1 and a_2.

3 From a fixed point theorem to a program

Given a context $(A^{(0)}, A^{(1)}, \rho)$, let $\mathbf{H} = \mathbf{H}(A^{(0)}, A^{(1)}, \rho)$ be the set of all pairs of extensive operators defined in the previous section. Similarly, the set of all pairs of closure operators will be denoted by $\mathbf{T} = \mathbf{T}(A^{(0)}, A^{(1)}, \rho)$. Then $\mathbf{H} = (\mathbf{H}, \leq)$ and $\mathbf{T} = (\mathbf{T}, \leq)$ are posets, $\mathbf{T}$ is a sub-poset of $\mathbf{H}$, and $\mathcal{G}, \mathcal{C} \in \mathbf{T} \subseteq \mathbf{H}$. Motivated by formula (1), we define a mapping $f : \mathbf{H} \to \mathbf{H}$, $\mathcal{D} = (\mathcal{D}^{(0)}, \mathcal{D}^{(1)}) \mapsto \mathcal{E} = (\mathcal{E}^{(0)}, \mathcal{E}^{(1)})$ by

$$\mathcal{E}^{(i)}(X) := \mathcal{D}^{(i)}(X) \cap \bigcap_{Y \in X\psi_i} \bigcup_{y \in \mathcal{D}^{(1-i)}(Y)} \{y\}\rho_{1-i} \,. \tag{2}$$

Clearly, $\mathcal{E} = f(\mathcal{D}) \in \mathbf{H}$, f is a monotone mapping, and $f(\mathcal{C}_{n+1}) = \mathcal{C}_n$ for all n. Since $f(\mathcal{D}) \leq \mathcal{D}$ for all $\mathcal{D} \in \mathbf{H}$, we will call f a *contraction map*. The following proposition is mentioned to shed more light on the topic only, and will not be used in the sequel.

Proposition 1. *Given a finite context $(A^{(0)}, A^{(1)}, \rho)$, $\mathbf{T} = (\mathbf{T}, \leq)$, the set of pairs of closure operators over $(A^{(0)}, A^{(1)}, \rho)$, is an (upper) semimodular coatomistic meet-semidistributive lattice, and $\mathbf{T}$ is closed with respect to the contraction map f.*

Proof. Let L_i be the poset of closure operators over $A^{(i)}$, $i = 0, 1$. Then, according to Corollaries 30 and 58 in Caspard and Monjardet [2], L_i is a lattice that has some nice properties, including those listed in the proposition. Notice that [2] attributes some of these properties to others, including Demetrovics, Libkin and Muchnik [5], Duquenne [6] and Ore [10]. Since $\mathbf{T}$ is the direct product of L_0 and L_1 and the properties we consider are clearly preserved by finite direct products, the first part of the statement is shown. The statement about the contraction map is included, modulo notational changes, in the proof of Lemma 1 in [3]. □

If $\mathcal{D} \in \mathbf{H}$ and $f(\mathcal{D}) = \mathcal{D}$ then $\mathcal{D}$ is called a fixed point of f. As usual, for $\mathcal{D} \in \mathbf{H}$ the set $\{\mathcal{E} \in \mathbf{H} : \mathcal{E} \leq \mathcal{D}\}$ will be denoted by $(\mathcal{D}]$. Since f is a monotone contraction map, $(\mathcal{D}]$ is always closed with respect to f. Remembering that $(A^{(0)}, A^{(1)}, \rho)$ is assumed to be finite, we have the following theorem.

Theorem 1. *$\mathcal{C}$ is the largest fixed point of f in $(\mathcal{G}]$.*

Proof. Since the context is finite, there is a k with $\mathcal{C}_k = \mathcal{C}_{k+1} = \mathcal{C}$. Hence $f(\mathcal{C}) = f(\mathcal{C}_k) = \mathcal{C}_{k+1} = \mathcal{C}$, so $\mathcal{C}$ is a fixed point of f. Clearly, $\mathcal{C} \in (\mathcal{G}]$.

Now let $\mathcal{D} \in (\mathcal{G}]$ be an arbitrary fixed point. Then, using that f is monotone, $\mathcal{D} = f(\mathcal{D}) \leq f(\mathcal{G}) = \mathcal{C}_1$. So $\mathcal{D} = f(\mathcal{D}) \leq f(\mathcal{C}_1) = \mathcal{C}_2$, etc. Thus $\mathcal{D} \leq \mathcal{C}_k = \mathcal{C}$. □

Even if the above theorem is a simple statement, it is useful from algorithmic point of view. The speed of the obvious algorithm for computing $\mathcal{C}$ depends on how fast the sequence $\mathcal{C}_0 = \mathcal{G}$, $\mathcal{C}_1$, $\mathcal{C}_2, \ldots$ decreases. If we follow what formula (1) says then we obtain $\mathcal{C}_{n+1}$ from $\mathcal{C}_n$ in two steps. In the first step we compute, say, $\mathcal{C}_{n+1}^{(0)}$

from $(\mathcal{C}_n^{(0)}, \mathcal{C}_n^{(1)})$, and then in the second step we compute $\mathcal{C}_{n+1}^{(1)}$ from $(\mathcal{C}_n^{(0)}, \mathcal{C}_n^{(1)})$. However, we could obtain a more rapidly decreasing sequence if we performed the second step from $(\mathcal{C}_{n+1}^{(0)}, \mathcal{C}_n^{(1)})$ instead of $(\mathcal{C}_n^{(0)}, \mathcal{C}_n^{(1)})$. (This would also mean less memory usage, which would save some additional time, too.) We will refer to this strategy as the *modified algorithm.*

Remark 1. The modified algorithm computes $\mathcal{C}$, and it is at least as fast as the straightforward algorithm suggested by formula (1). (In fact, it is usually faster.)

Indeed, for $i \in \{0,1\}$ we define a mapping $f_i : \mathbf{H} \to \mathbf{H}$, $(\mathcal{D}^{(0)}, \mathcal{D}^{(1)}) \mapsto (\mathcal{E}^{(0)}, \mathcal{E}^{(1)})$ such that $\mathcal{E}^{(i)}$ is defined as in formula (2) and $\mathcal{E}^{(1-i)} = \mathcal{D}^{(1-i)}$. (It follows easily from Proposition 1 that $\mathbf{T}$ is closed with respect to the contraction maps f_i, $i \in \{0,1\}$, but we do not need this fact in the proof.) The modified algorithm produces the sequence

$$f_0(\mathcal{G}),\ f_1(f_0(\mathcal{G})),\ f_0(f_1(f_0(\mathcal{G}))),\ f_1(f_0(f_1(f_0(\mathcal{G})))),\ \ldots\ .$$

Computing two new members of this sequence needs a slightly less computer work than computing one new member of the sequence $\mathcal{C}_0 = \mathcal{G}$, $\mathcal{C}_1$, $\mathcal{C}_2$, ... Hence all we have to show is that, for every n, the $2n$-th member of the first sequence is above $\mathcal{C}$ and below $\mathcal{C}_n$. But this follows via a trivial induction, since $f(\mathcal{D}) \leq f_i(\mathcal{D}) \leq \mathcal{D}$ and $f(f(\mathcal{D})) \leq f_1(f_0(\mathcal{D})) \leq f(D)$ hold for all $\mathcal{D} \in \mathbf{H}$.

It is clear from the proof of Theorem 1 and the argument following Remark 1 that instead of the poset $\mathbf{H}$ we could have worked only with the lattice $\mathbf{T}$. However, the advantage of $\mathbf{H}$ is not only to make Theorem 1 stronger. In a practical calculation, like computing $\mathcal{C}^{(i)}(X)$ just for a single X, it gives a better theoretical background: we can reduce the $\mathcal{G}^{(i)}(Y_j)$'s for certain (not necessarily distinct) subsets Y_j of $A^{(0)}$ and $A^{(1)}$ according to (2) in an arbitrary order, and we do not have to care if the actual pair of operators is a pair of closure operators, the process converges to $\mathcal{C}$.

The algorithm given by Remark 1 was developed into a modest computer program which computes $\mathcal{C}$. The program is available at the author's home page. (Although the source code is in Borland's old Turbo Pascal 7.0 for MS DOS, the executable version runs in today's Windows environment as well.) When $|A^{(0)}|$ or $|A^{(1)}|$ is large then it is not possible to determine $\mathcal{C}$ in the practice, at least not with *this* program, for we would need $2^{|A^{(0)}|} + 2^{|A^{(1)}|}$ steps even to store $\mathcal{C}$. However, as it is clear from formula (1), we can determine $\mathcal{C}^{(i)}(X)$ for all X with $|X| \leq m$ and all $i \in \{0,1\}$ without determining the whole $\mathcal{C}$, and this is much faster when m is not too large. The program allows $m \in \{2,\ldots,9\}$ when $|A^{(0)}|, |A^{(1)}| \leq 14$, and it allows only $m = 2$ when $|A^{(0)}|, |A^{(1)}| \leq 48$. However, the running time even for a single context with $m = 9$ and $|A^{(0)}| = |A^{(1)}| = 14$ is usually too long to wait for. The program can generate and test many random contexts. The experimental results obtained by the program are reported in the following section.

4 Experimental results obtained by the program

The computational results will be given by tables, which are followed by the explanations of their concise notations. Even if these results are not exact mathematical theorems, they shed some light on the goal of the paper, and they may induce exact numerical bounds or asymptotical statements in the future. Notice that when only some specific relations (e.g., partial orders) are considered then the frequency of $\mathcal{C} \neq \mathcal{G}$ can be quite different from what we can learn from the present section, cf. [4].

size	4	5	6	8	10	12	14	20	30	40	48
$\lvert\{\text{tests}\}\rvert$	1000	1000	1000	100	100	100	100	100	100	100	100
$(-,-,-)$	549	757	889	98	100						
$([0,4],-,-)$	549	757	889	98	100	100	100				
$([0,2],-,-)$	535	707	844	97	100	100	100				
$(\{2\},-,-)$	282	517	736	94	100	100	100	100	100	100	100
$(\{2\},\subset,-)$	235	484	721	94	100	100	99	86	18	4	1
$([2,\infty),\subset,\neq)$	0	134	491	96	100						
$([2,4],\subset,\neq)$	0	134	491	96	100	100	99				
$(\{2\},\subset,\neq)$	0	134	470	92	100	100	99	86	18	4	1

Table 2

In Table 2, "size" denotes $|A^{(0)}| = |A^{(1)}|$, i.e., only "square" contexts have been tested. The number of contexts tested with the given size is denoted by $|\{\text{tests}\}|$. For a given context, $\mathcal{C} \neq \mathcal{G}$ can be due to some more or less trivial reason like $\mathcal{C}^{(i)}(\emptyset) \neq \mathcal{G}^{(i)}(\emptyset)$. Therefore the program counted those contexts for which there is an $i \in \{0,1\}$ and an $X \in P(A^{(i)})$ such that $\mathcal{C}^{(i)}(X) \neq \mathcal{G}^{(i)}(X)$ and X satisfies some further conditions. Each of these further conditions is denoted by a vector (α,β,γ). Here α is missing or it is a set of integers, like $[2,4] = \{2,3,4\}$. If α is a set of integers then $|X|$ has to belong to α. If β not missing then it is the "$\subset$" sign and X has to satisfy $X \subset \mathcal{C}^{(i)}(X)$. If γ is not missing then it is the "$\neq$" sign and X has to satisfy $\mathcal{G}^{(i)}(X) \neq A^{(i)}$. For example, the row $(-,-,-)$ gives the number of contexts with $\mathcal{C} \neq \mathcal{G}$, and the entry 484 means that among 1000 random contexts there are 484 contexts containing a 2-element subset X with $\mathcal{C}^{(i)}(X) \neq \mathcal{G}^{(i)}(X)$ and $X \subset \mathcal{C}^{(i)}(X)$.

It is important to emphasize that, for each column, the program produced the given number of random contexts first, and counted those context that have the desired property only afterwards. In other words, different entries in the same column refer to the same set of random contexts. Due to the limited power of the program some entries are missing, but some obvious relations among the numbers in the same column give lower bounds for the missing entries.

We may also ask the question that if we take a random context table of size $n \times n$ and choose an $i \in \{0,1\}$ and a subset X of $A^{(i)}$ randomly then what is the chance that $(-,-,-)$: $\mathcal{C}(X) \neq \mathcal{G}(X)$, or $([2,\infty),\subset,\neq)$: $2 \leq |X|$ and $X \subset \mathcal{C}^{(i)}(X) \neq$

$\mathcal{G}^{(i)}(X) \neq A^{(i)}$. The experimental results for some values of n are reported in Table 3.

size	3	4	5	6	7	8	9
$\lvert\{\text{tests}\}\rvert$	1000	1000	1000	1000	1000	1000	1000
$(-,-,-)$	17	39	82	185	306	402	571
$(\{2\},\subset,\neq)$	0	0	0	12	35	54	86

Table 3

Table 2 gives the strong belief[3] that a "medium sized" square context gives an "essentially new" $\mathcal{C}$ with high probability. Here "essentially new" means that the condition $(\{2\},\subset,\neq)$ holds for some X. However, this probability decreases rapidly when the size of the context grows.

Let us call a random context a *p-random context*, $0 < p < 1$, if we put a cross to each entry with probability p, independently from other entries. So far we have considered 0.5-random contexts. However, we may get different results with other values of p. For example, we tested 100 p-random 40 × 40-sized contexts with different values of p, and counted the essentially new contexts among them (in the sense of $(\{2\},\subset,\neq)$). The result is given by Table 4.

p	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
$(\{2\},\subset,\neq)$	100	68	14	5	2	3	23	64	77

Table 4

Finally, we tested some contexts from the real life: essentially all those contexts from Ganter and Wille [7] which are given by simple context tables (with × being the only entry) and whose size fits into the program. (Sometimes the context was given by a multi-valued table and we had to reduce it.) Each column represents a context, and the column label tells us which context of [7] is considered. "Yes" for a context means that $\mathcal{C}$ is distinct from $\mathcal{G}$ and the condition (α,β,γ) given in the row label is satisfied.

	1.1	1.5a	1.5b	1.13	1.16	1.21	1.23	1.24	2.4	2.13	2.15
$\lvert A^{(0)}\rvert$	8	8	5	5	14	6	6	8	7	12	14
$\lvert A^{(1)}\rvert$	9	5	4	25	16	12	8	8	7	9	9
$(\{2\},\subset,\neq)$	no	no	no	yes	no	no	yes	no	yes	no	yes
$([0,6],-,-)$	yes	no	no	yes		yes	yes	no	yes	no	yes

Table 5

[3]Theoretically there could be some unknown hidden connection between $\mathcal{C}$ and the built-in random number generator and this could mislead us, but the chance of this is minimal.

References

[1] Agarwal, R., Imielinski, T., and Swami, A. Mining association rules between sets of items in large databases. in *Proceedings of the 1993 ACM SIGMOND international conference on Management of Data (SIGMOND'93)*, pages 207-216, ACM Press, May 1993.

[2] Caspard, N., and Monjardet, B. The lattice of closure systems, closure operators, and implicational systems on a finite set: a survey. *Discrete Applied Mathematics* 127(2):241–269, 2003.

[3] Czédli, G. 2-uniform congruences in majority algebras and a closure operator. *Algebra Universalis* 57(1):63–73, 2007.

[4] Czédli, G. A stronger association rules in lattices and posets, *Order*, submitted.

[5] Demetrovics, J., Libkin, L.O., and Muchnik, I.B. Functional dependencies in relational databases: a lattice point of view. *Discrete Applied Mathematics* 40(2):155–185, 1992.

[6] Duquenne, V. The lattice of all 1-meet-subsemilattices of a finite lattice. 1987 (unpublished notes).

[7] Ganter B., and R. Wille, R. *Formal Concept Analysis: Mathematical foundations.* Springer-Verlag, Berlin — Heidelberg, 1999.

[8] Lakhal L., and Stumme, G. Efficient mining of association rules based on formal concept analysis. in *Formal Concept Analysis*, Lecture Notes in Computer Science, vol. 3626/2005 (editors: B. Ganter, G. Stumme and R. Wille), pp. 180–195, Springer-Verlag, Berlin — Heidelberg, 2005.

[9] Polster, B. *The mathematics of juggling* Springer-Verlag, New York, 2003.

[10] Ore, O Combinations of closure relations. *Ann. Math.* 14:514–533, 1943.

[11] Wille, R Restructuring lattice theory: an approach based on hierarchies of concepts. *Ordered sets* (Banff, Alta., 1981) *NATO Adv. Study Inst. Ser. C: Math. Phys. Sci.* **83**, pages 445–470, Reidel, Dordrecht-Boston, Mass., 1982.

Received 24th September 2007

Acta Cybernetica **19** (2009) 159–175.

Filter Bank Design for Melody Recognition

Zoltán Gera*

Abstract

Recognizing different features of a waveform to later recompose the music that was originally present in the signal is a difficult task. There are numerous fields of application where these techniques are known to be useful including music authoring, digitizer design, automatic music transcription. There are many different methods that can be used for this purpose giving somehow inadequate quality regarding noise, polyphony or time- / frequency localization compared to the human auditory system. In this article, I will show a new filter design method specifically designed to be aware of human perception features. I will also show the way how a complete filter bank can be assembled and used for melody recognition in real time. Finally, I will point out the benefits of this filter design compared to other methods.

Keywords: DSP, melody recognition

1 Introduction

1.1 Main Objectives

Melody recognition is a process where different features are extracted from a waveform to later form music data. These features, such as *pitch*, *note length* and *loudness*, are high-level, subjective and abstract psychological perceptions [9] that are hard to deal with. Recognizing rhythm and melody simultaneously should involve an analyzation process both in time- and frequency domains [14]. This resolution should have special requirements compared to other conventional signal processing techniques. Examining these requirements makes us possible to reach superior quality over standard methods [1, 15, 20].

Time resolution must have an adequate *separation* property. Separation avoids blurring sounds together that were audible as two distinct notes. It should also give *continuity* in a way that it does not leave short but significant sounds out of processing. These requirements, together with their exact parameters, can be derived from psychoacoustic measurements [9].

The frequency resolution should fit the exponential *scale of music* [13]. It should give correct state information about every note along the scale regarding the note

*ELTE IK, Budapest, E-mail: gerazo@elte.hu

is sounded or not. Neighboring notes should be clearly distinguishable from each other. Every sound that has significant harmonic content should be classified along the *melodic scale* as one note. These requirements are related to music theory [13] and psychoacoustics [1, 9, 15].

Our computational load expectancies aim not less than real-time performance. Running the recognition process in real time is a key to give us numerous new ways of utilization creating revolutionary techniques in digital music authoring.

1.2 Idea of Filter Bank

Every note of the melodic scale is now considered as a frequency interval or *channel*. We should create individual filters which fulfill our time domain requirements at one particular channel. These filters do their work with their own channel data only, they do not interfere with others, so the whole bank will also fulfill the frequency domain requirements as well. It is easy to find out that these filters should be *bandpass filters* with a well-defined *passband*. Having the scale of western music [13] where an *octave* difference is a multiplication by 2, an octave has 12 different notes and a convention exists where A-4 note means frequency $440Hz$, we get the following center frequencies for notes:

$$440 \cdot 2^{n/12} \tag{1}$$

where n is the note number relative to A-4. We want to have whole octaves in our range of investigation, so the interval from C-1 to B-9 is $n \in [-45..62]$ which covers the full audible frequency spectrum. That is only 108 filters with the following passbands:

$$\left[440 \cdot 2^{(2n-1)/24}, 440 \cdot 2^{(2n+1)/24}\right] \tag{2}$$

Every filter will be used to measure note state on a channel at a particular point of the input signal. Note state comes from a measurement that gives us some kind of value connected to *perceptual loudness* [9]. The time interval of different measurements should be short enough to fulfill separability criteria. This interval also depends on frequency. The higher the pitch is, the shorter the interval should be. (simple outcome of period length) However, a filter need not to be used at every point of the input data because far less work also gives adequate time domain resolution [8]. It would be also a waste of processing power. This means that only FIR (*Finite Impulse Response*) filters are good for this purpose because IIR (*Infinite Impulse Response*) filters can only be computed along an interval, rather than at a single point, because of their recursive nature. Filter length and every other parameter should be individually computed for every note, and separate filters should be designed for all notes.

Filters should be created to give constant quality on every channel, so delivering the same attenuation features with every note. This idea has the same concept as the classic method called *constant-Q transform* (CQT) [18]. However, CQT and all derived methods [3] lack general using of psychoacoustic principles apart from the fact that they also operate with non-linear scale. This is the explanation why CQT

can be computationally more intensive than FFT, even if CQT produces less data and lacks reversibility features of FFT [18]. Our constructed filters will form a *filter bank*. Under certain circumstances, the bank can produce analysis with adequate quality. I will show the details in the next section.

2 Filter Design

2.1 Creating a Filter

We need a specifically designed bandpass filter which attenuates (suppresses) everything outside passband, but gives perceptual loudness inside. According to the experiments of Fletcher and Munson [5], *equal-loudness contours* (or *Fletcher-Munson curves*) show the characteristics of how the human auditory system responds to different sound pressure levels at different frequencies. To have our filter produce perceptual loudness values rather than absolute levels, the filter's passband should fit the reciprocal of the mentioned curves called *weightings*. Different weighting functions do exist. We will use *A-weighting* in our work because it is designed to work well on harmonic waveforms. Other weightings perform better on noise-like waves. They could be used for special, rhythm intensive recognition tasks.

Using a weighting function to get perceptual loudness instead of simple volume level is a key to be able to later compare different peaks and choose dominant harmonics. It is quite common that the detection or transcription of various instruments give false results because the underlying algorithm uses absolute measurements and direct physical values when dealing with abstract, cognitive parameters. There are more good approximations of the A-weighting. We will use one of them. The ideal filter for one channel and the A-weighting is shown in Figure 1.

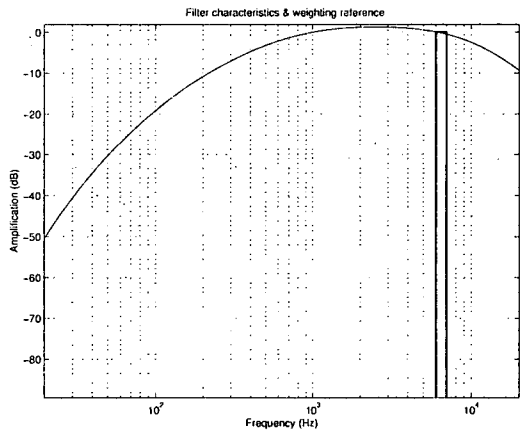


Figure 1: Ideal filter and A-weighting in frequency domain

Creating the impulse response of this filter is easy for any arbitrary note n.

To realize this representation on computer, we choose a number l_d as the *design length* of the filter. Because the filter designing is an off-line procedure (no real-time requirements), we choose l_d to be large. The filter representation is

$$f(x) = \sum_{i=\lfloor l_d 440 \cdot 2^{(2n-1)/24}/s \rceil}^{\lfloor l_d 440 \cdot 2^{(2n+1)/24}/s \rceil} \cos\left(\frac{x \cdot 2\pi i}{l_d}\right) \cdot 10^{\frac{w_A(i/l_d \cdot s)}{20}} \tag{3}$$

where f is the vector with length l_d, $x \in [0..l_d - 1]$, the $\{\ldots\}$ operator rounds to the nearest integer, w_A is the A-weighting function and s is the *sample rate* in Hz which defaults to 44100. The multiplication after the cosine applies the weighting function to the result.

One interesting feature of the resulting vector that its significant coefficients are all near the two ends. It can be proven that whenever we add cosines with near frequencies, specially frequencies from a narrow interval, cosine functions are always in phase at the beginning and end of the interval giving the biggest coefficients there. We change the phase of the resulting vector to have the largest coefficients at the center of the vector.

$$f' = \left[f\left(\left\lceil\frac{l_d}{2}\right\rceil\right), \ldots, f\left(l_d - 1\right), f\left(0\right), \ldots, f\left(\left\lceil\frac{l_d}{2}\right\rceil - 1\right)\right] \tag{4}$$

This allows us to later apply window functions with smaller information loss. The impulse response of this quasi-ideal filter for note n is shown in Figure 2. (The filter is quasi-ideal, because it is discretized now, but is still in long-length representation.)

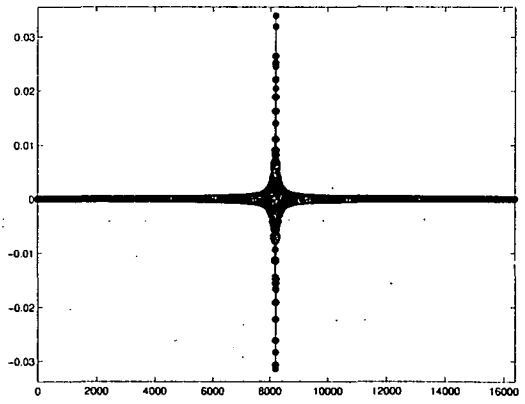


Figure 2: Quasi-ideal filter in time domain

The quasi-ideal filter has still perfect characteristics, but it is far too long to be used in real time. We need to chop insignificant parts off the vector to have a short and real-time applicable filter. Fortunately, we can do this because significant coefficients are in the center of the vector.

Chopping is best done through applying a *window function*. If we use a compact function for this purpose, we can keep the good features previously mentioned. Applying a *window function* on a filter can have further positive effects on the final result. Our goal is to reduce the power of crosstalk coming from outside the passband, so using *Kaiser window* which maximizes the ratio of the mainlobe energy to the sidelobe energy seems to be a good choice. Later we will also determine the parameters of this window. If $g(x)$ is the final filter with length l_f (a parameter to be discussed later), $f'(x)$ is our current filter with length l_d (design length), the window function is $w(x)$ also l_f long, then chopping and applying the window function on filter $f'(x)$ is simply the following

$$f'' = \left[f'\left(\left\lceil \frac{l_d}{2} \right\rceil - \left\lceil \frac{l_f}{2} \right\rceil + 2 \right), \ldots, f'\left(\left\lceil \frac{l_d}{2} \right\rceil + \left\lfloor \frac{l_f}{2} \right\rfloor + 1 \right) \right] \tag{5}$$

$$g(x) = f''(x) \cdot w(x) \tag{6}$$

Filter design through manipulating different window functions is told to be an art, because many times, there are no direct methods of getting the best values. Several experiments are required to find a close-to-the-best solution. This is definitely our case. The previously mentioned windowing step uses a window that is a combination of a standard rectangular window (for chopping) and a Kaiser window (for tuning up characteristics). It is possible to further improve our filter with more window functions combined into our windowing step. No matter how fancy our window composition is, this windowing step is done in one step only at design time. Figure 3 shows the impulse response of the filter after the windowing step.

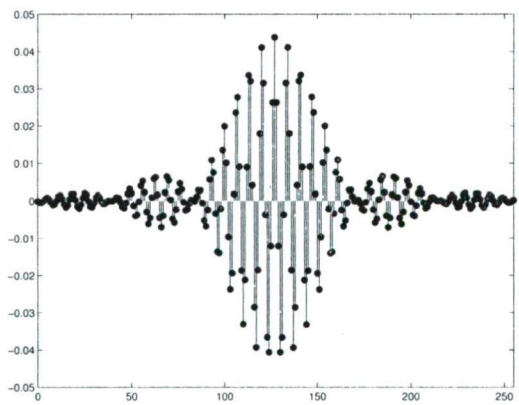


Figure 3: Final impulse response of the filter

Finally, we should normalize the passband response of the filter to have exactly the same attenuation as the A-weighting has at the center frequency of the passband. The easiest way to do this is to use the filter on a sine wave of center

frequency, and measure the SPL (*Sound Pressure Level*) of the resulting wave to have the correct coefficient to be multiplied on the filter. This way, we have gained the desired response properties implicitly compensating the power distortion effect of the windowing step and the filter length difference between different channels.

$$t(x) = \sum_{i=0}^{l_f-1} \cos\left((x-i)\cdot 2\pi \cdot \frac{440\cdot 2^{(2n-1)/24} + 440\cdot 2^{(2n+1)/24}}{s}\right)\cdot g(l_f - i) \quad (7)$$

where g is the unnormalized filter, l_f is the final filter length and the length of g, t is the filtered wave. The length of t should not be too small to have adequate precision. Let this length be l_p (we are still in design time), so variable will go $x \in [0..l_p - 1]$. The SPL of t is the quadratic mean

$$SPL = \sqrt{\frac{\sum_{i=0}^{l_p-1} t(i)^2}{l_p}} \quad (8)$$

so the final filter h will be g multiplied by a constant:

$$h(x) = g(x)\cdot \frac{\sqrt{\frac{1}{2}}\cdot 10^{\left(w_A\left(\frac{440\cdot 2^{(2n-1)/24}+440\cdot 2^{(2n+1)/24}}{2}\right)/20\right)}}{SPL} \quad (9)$$

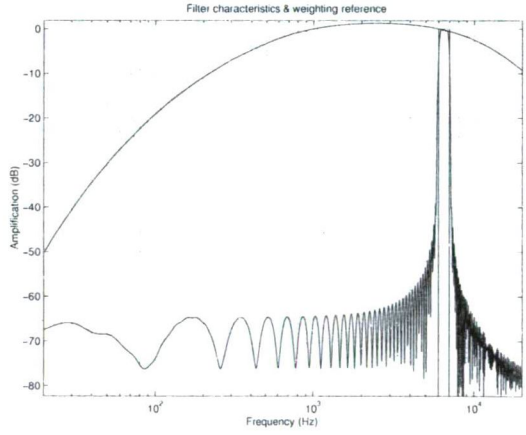


Figure 4: Frequency response of the final filter

Figure 4 shows the frequency response of the final filter. The ripples are the results of giving the filter a short discretized form to fit in a short vector. This is the cost of filtering real-time compared to the ideal filter world.

2.2 Evaluating the Design

Our choice was the Kaiser window as a window function. We can use different β parameters for the window. Zero equals the rectangular window. There are some experiments in Table 1 where different window parameters were used with our design method.

Increasing β lowers sidelobes which reduces crosstalk from stopband, but widens the mainlobe giving smoother transition between stopband and passband. In classical filter design, wider mainlobe is to be avoided. For us, wider mainlobe is not as harmful as crosstalk, because its effects can be reduced by post-processing between neighbors on the final data. This also happens in the human auditory system where auditory nerves apply inhibition on neighboring nerves and cells [15, 16]. This choice is also a key to mimic perceptual functions better.

Evaluating the filter is important. We should know how good it is, what quality does it have. Evaluating a filter is done in frequency domain, in our case, with a logarithmic scale in both directions (frequency and amplitude level). Our design assures good approximation of the weighting function in the passband even in the worst case, so evaluating is mostly necessary in the stopband.

An error function will be created to rate the filter. The stronger the attenuation is in the stopband, the smaller error value the filter will get. There should be an attenuation threshold called the *ideal attenuation* which is adequate to our needs. Beyond this threshold, no error value is given for the filter to the specific frequency point. The errors of different frequency points are cumulated to form the final error value for the filter.

$$e = \sum_{i=1 \& a(i) > a_i}^{p} (a(i) - a_i) \tag{10}$$

where a is the frequency response of the filtering with p precision (length) having the attenuation values for different frequencies. These can be calculated the same way we did at the normalization procedure. The spectrum is calculated only for the audible interval, namely $20Hz - 20kHz$. Coefficient a_i is the ideal attenuation. Setting it to a reasonable value like constant 60 dB gives good results.

As was mentioned before, our design ensures good features in the passband. The frequency response of the whole stopband is quantifically interesting and we use it in our error function. But the accurate frequency response of the stopband has no importance.

This is not the case with the passband. The passband will let through peaks of the original signal which should be later compared to each other perceptually in both time- and frequency domains. The passband for this reason should be strictly fit to the weighting function. This is not a general criterion for all the passbands of different channels, but should specifically apply to all the individual passbands on their own. This causes that even the smallest ripples are not allowed inside the passband. Otherwise, small vibratos or other artistic techniques could cause level change in the output of a filter which was not present in the original signal at all.

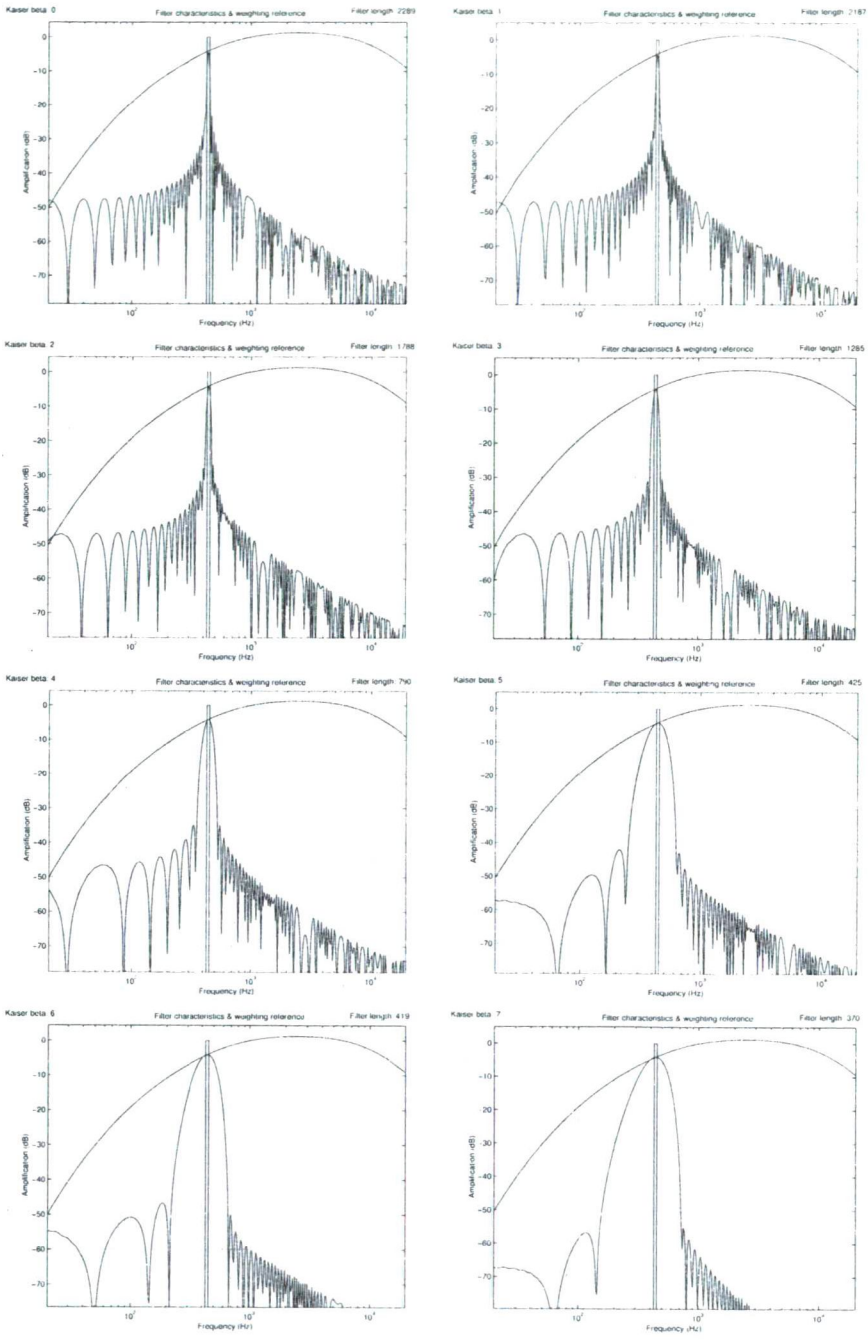


Table 1: Filter design with different Kaiser β parameters

This is a key feature to have a stable and steady output when the note is inside the channel of concern.

The conclusion is that comparing our filter design to other filter design methods has little use. The error function itself also does not serve this purpose. For example, the *Parks-McClellan optimal equiripple* filter design method reduces the number of filter coefficients by introducing ripples in the passband. While this technique turns out to be quantifically better than ours under normal circumstances (evaluating error along a linear scale in both passband and stopband), our method is superior with the mentioned special error function (evaluating along logarithmic scale and only in stopband). Our method introduces ripples only in the stopband reducing the number of coefficients without disturbing the important features gained in the passband.

The length parameter of our filter design is still undetermined, so we get basically random error values after evaluating. Our error function will be used in the following to create an algorithm to determine the optimal filter for every note. This way, we will gain good performing filters on all channels specifically tailored to our needs.

3 Filter Banks

3.1 Filter Bank Assembly

A filter bank consists of many filters. It may contain more than one filter for a note. Our filter design method has many different parameters. We have determined what parameters have optimal values independent of frequency, so these parameters can be the same when designing filters that belong to different notes. However, the most important parameter, the final filter length is still a question. Figure 5 shows different quality measurements of a design of every filter along the scale using constant length parameters.

Obviously, the final filter length parameter cannot be a constant. It will always depend on the note we are designing. The quality measurement of a filter introduced in the previous section is a good point to start from. We should design filters for every note with constant quality (constant error value). We need an algorithm to locate an optimal filter length parameter for every note. Optimality is reached here when the resulting filter's error value is below a preset threshold, and the filter is as short as possible. The pseudo-code of this process should be something like this:

1. Set parameters to their defaults and set length parameter to a really small value
2. Design the filter with current parameters
3. Evaluate the filter (using the error function)
4. If the error is below the quality threshold, but it is close to it, stop procedure

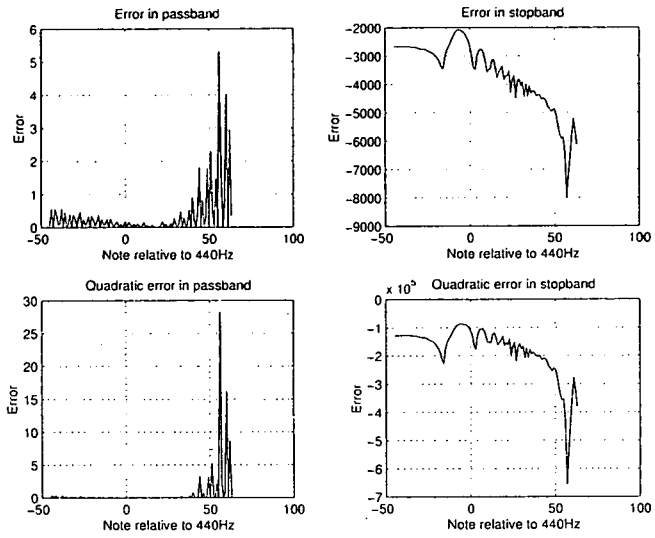


Figure 5: Filter evaluation along the full scale with constant length parameters

5. If the error is smaller then the quality threshold, decrease length parameter by a small amount, go to step 2

6. If the error is bigger then the quality threshold, increase length parameter by a bigger amount, go to step 2

As we have seen, the error values are not monotonic in the function of length. However, the trend of the function is monotonic, only small ripples cause the loss of strict-sense monotonicity. The above algorithm can be implemented as a modified logarithmic search, so determining the exact step amount at the increase/decrease phases is easy. (This exact step amount will be reduced as we get closer according to the original logarithmic search.)

Because this filter design method is not optimized for real time at all (it will run in design time), a search where the design and evaluation phases are repeated many times can be quite time consuming. We do not need to have the best parameter, we only need to get a close approximation. That's the reason of the uncommon stop condition in step 4. The logarithmic search will find a near optimal length value. Choosing smaller steps can improve quality and degrade performance. The bigger step toward longer filters favors quality over length on the long run.

The whole procedure finding optimal filters along the scale with this search algorithm using various error thresholds, extreme small step values and really close stop condition (to reach the ideal optimum and to be able to see the best result with worst performance) runs only some hours long on a common computer (test was done on Celeron 900MHz) in Matlab. This covers running the search algorithm for every channel along the scale multiple times to gain filters for all channels with

different quality parameters (5 filters per channel in this experiment). Because the Matlab testbed was using only basic calculations (own DFT to be able to modify also some parameters inside), recoding the whole procedure in a real programming language should give a drastic speedup to the process. (The same can be run under less than a minute.) Figure 6 shows the lengths of the resulting filters along the scale.

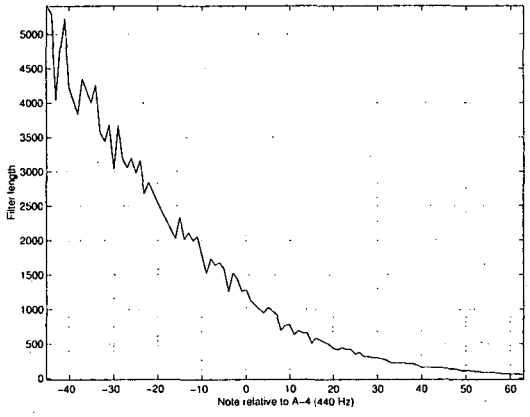


Figure 6: Lengths of filter results

3.2 Filter Bank Usage

Let vector f be the final filter of length l_f for a specific note, and let x be the input waveform. The filtered wave y is then

$$y(i) = \sum_{j=0}^{l_f-1} x(i - l_f + 1 + j) f(j) \tag{11}$$

Our task is to determine the loudness level of the original signal in different points of time via calculating the SPL of y at some points using the smallest number of calculations possible. The above equation allows finding values of y in one arbitrary point. We know that y only contains sines around the center frequency of the note that belongs to the filter. The basic unit of our real-time calculations, the *perceptual loudness measurement* at point i for the filter of note n with s as sample rate is as follows.

$$z(i) = \sqrt{y\left(i - \left\lceil \frac{s}{440 \cdot 2^{n/12}} \cdot \frac{1}{8} \right\rceil\right)^2 + y\left(i + \left\lfloor \frac{s}{440 \cdot 2^{n/12}} \cdot \frac{1}{8} \right\rfloor\right)^2} \tag{12}$$

This is a straightforward consequence of the equation $\sin^2 x + \cos^2 x = 1$.

This measurement is only accurate when we measure a signal containing only the center frequency. Other frequencies can also be present in y. These are frequencies near the center frequency, because we are using a well-defined passband filter. Taking the center as unit, these are $2^{-1/24} \ldots 2^{1/24}$ away from the frequency of center which means $-2.85\% \ldots 2.93\%$ fluctuation in frequency. This maximum 3% fluctuation level gives the measurement inaccuracy $\sin^2 x + \cos^2 (x + 2\pi \cdot 0.03)$ under the period of the center frequency shown in Figure 7.

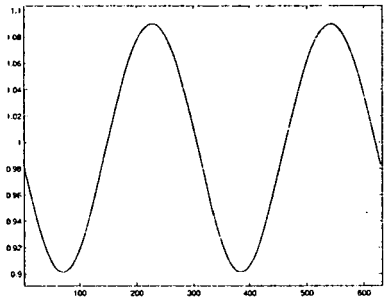


Figure 7: Perceptual loudness measurement inaccuracy through 1 period of center frequency

The fluctuation of inaccuracy is two times faster then the change of the waveform level itself, so it is enough to do the loudness measurement once per period changing the exact phase a bit randomly, and average the last measurements to suppress the fluctuation and get the real perceptual loudness level. According to psychoacoustic considerations, we even do not have to do the measurement once per period. It is adequate to do it rarelier, but still the random phase change and averaging logic should be used.

Summing up the consequences, we only have to do the loudness measurement rarely, at most once per period (or rarelier if psychoacoustic features allow this). This only takes using the filter twice to calculate two distinct points of y. The function y is never fully computed. The random phase change in choosing the discrete points of y to be computed is not a simple smoothing technique. It is necessary to compensate for slightly out of tune instruments, but first of all, to compensate for inharmonic content of instruments. Dealing with inharmonic content is the most important factor of this kind of smoothing, because inharmonicity is there even in the most harmonic real-world instruments. Inharmonicity is also an organic part of rhythm instruments.

It is obvious, that the measurement rate is proportional to the center frequency, so the note to which the filter belongs. Figure 8 shows the ratio between the filter length and the period length of the center frequency.

This result is really interesting, because not only does it show that the longer filter length at low frequencies are compensated by the fact that these filters should be applied less, but it also turns out that using the method is cheaper on lower

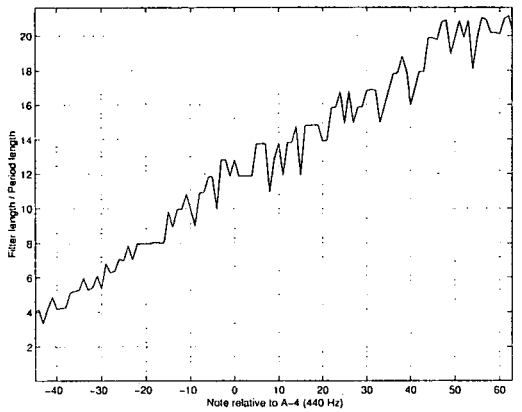


Figure 8: Filter length per period length ratio on scale

notes.

It is possible to construct more filter banks with different quality threshold values. This would give the possibility to the real-time method to use extra computational power on better machines, and still give adequate results on low-end ones. It is also possible to dynamically and adaptively change the used filter bank to have good time- and frequency resolution as well. This trick can be used to virtually get rid of the Heisenberg uncertainty principle similar to what happens in the human brain as supposed by biologists [17].

3.3 Comparing to Other Methods

Finally, we do some comparisons. First, we compare our method to a trivial solution of our problem that uses FFT. Special comparing is needed in case of real-time applications. A method running in real time is better described by the number of operations per second than the asymptotic number of operations in the function of input data (especially because the input rate is constant). However, our calculations will still be estimations under certain circumstances. We will investigate the basic case of the two algorithms, but we will focus on comparing the two with parameters having similar quality features. This quality will be really high to be able to see the significant differences, if there are any.

We start with our method. Using the filter on one point according to

$$y(i) = \sum_{j=0}^{l_f-1} x(i - l_f + 1 + j) f(j) \tag{13}$$

takes

$$2 \cdot l_f \tag{14}$$

operations. A perceptual loudness measurement at a point according to

$$z(i) = \sqrt{y\left(i - \left\lceil \frac{s}{440 \cdot 2^{n/12}} \cdot \frac{1}{8} \right\rceil\right)^2 + y\left(i + \left\lfloor \frac{s}{440 \cdot 2^{n/12}} \cdot \frac{1}{8} \right\rfloor\right)^2} \tag{15}$$

takes

$$4 \cdot l_f + 8 \tag{16}$$

operations.

Figure 6 gives us the exact $l_f(n)$ filter length values for every note. We also know the center frequency $f_c(n)$ of note n. For simplicity, we use only one filter bank, but we make a measurement on every period of the center frequency. This gives the total number of operations per second as follows:

$$\sum_{n=-45}^{63} f_c(n) \cdot (4l_f(n) + 8) \approx 250,475,000 \tag{17}$$

This 251 million operations per second need a high-end machine to accomplish. Of course, the 60dB attenuation threshold can be lowered to give still adequate results with relaxed computational needs.

Now, we create a similar method using FFT to reach the same high quality resolution. We use a simple method of FFT overlapping windows and calculate the number of operations for this method. For simplicity, we omit the use of window functions (which is obviously a cheat to help FFT).

The center frequency of the two lowest notes are $32.703Hz$ and $34.648Hz$, therefore the minimal required resolution of FFT is approximately $2Hz$. At sample rate $44100Hz$, this means a window length of 22050. The first applicable window length for FFT is therefore $2^{15} = 32768$. Using conventional radix-2 FFT with bit reversal in the beginning, the number of operations for one FFT is

$$32,768 + 15 \cdot (5 \cdot 32,768) = 2,490,368 \tag{18}$$

To reach the desired time resolution as well, separate FFTs should be calculated according to the center frequency of the highest note. This is $16,744Hz$, therefore the total number of operations per second is

$$2,490,368 \cdot 16,744 = 41,698,721,792 \tag{19}$$

and the ratio between the two methods is

$$\frac{41,698,721,792}{250,475,000} \approx 166 \tag{20}$$

Our algorithm is significantly faster than trivial FFT algorithm for the same task. In the investigation above, we have set all parameters to get the same quality from both methods. FFT turned out to be much slower. This also means, that with the same performance (same number of operations), FFT gives much lower quality

(which is inadequate for our special task) or on the other hand, our new algorithm delivers far higher quality than FFT with the same number of operations.

Our special requirement is a frequency resolution using logarithmic scale. Using wavelet-like methods introduced in [14] would give high performance with extreme poor quality, because the octave frequency resolution is far not enough for a 12-degree scale (Wavelet transforms are reversible, so dividing an octave into 12 parts would be a waste according to the wavelet philosophy).

Methods that use linear scale can deliver adequate quality, but perform worse than ours. This means lower quality with same performance (number of operations), or much lower performance with the same quality, simply because linear scale does not fit our task. This is the case using FFT (as shown above) and FFB according to [3]. FFB (*Fast Filter Bank*) uses basically the same frequency resolution as FFT does, but uses more sophisticated filters for tuning up individual channel characteristics. The cost is also more operations per second which further lowers performance.

The original CQT introduced in [18] uses pure logarithmic frequency resolution fit to the music scale. The strange property of CQT is that it needs far more computations than FFT according to the original [18] results. There are more papers about faster methods of calculating CQT, the most recent is [3], but none of them could be faster than FFT. This is the simple result of CQT's original concept, that it wants to deliver continuous signal output as FFT does. While this can be useful for some special off-line analyzation purposes (examining classical big orchestral music in printed paper), it was shown above that general melody recognition does not need this feature as the human brain also lacks this huge precision of resolution. My method can be faster than CQT only because I use psychoacoustic features which are not used by previous works.

The other remarkable result of [3] is the CQFFB (*Constant Quality Fast Filter Bank*), which also uses the filter tuning trick on CQT to deliver better separation properties. While this further lowers performance but raises quality compared to CQT, the main problem with this is that the individual filters are still not constructed according to the features of music perception. They rather form a consistent bank of filters with similar passband limits, but passband frequency responses are not designed intentionally. CQFFB also lacks deep psychoacoustical design so inherits the weaknesses of CQT regarding low performance (high number of operations).

It should be also mentioned that both CQT and CQFFB [3] methods have accelerated versions (BQT and BQFFB). These *bounded-Q* methods gain performance by using only one bank for an octave instead of using banks for all channels as the original versions did. This trick makes the frequency separation worse. Different notes have slightly different filters and passbands (because inside an octave, the frequency resolution is linear), so it is impossible to design filters with as strict criteria as we did. My method designs filters specifically for all individual notes, and only one filterbank is constructed which can be applied much faster delivering the expected frequency responses.

References

[1] Aslan, A. Music Perception as a Topic of Cognitive Psychology. Doguş Universitesi Dergisi. 8(2):117-127, 2007.

[2] Bohn, D. *Audio Specifications.* Rane Corporation, 2000.

[3] Diniz, F. C. C. B., Kothe, I., Netto, S. L., Biscainho, L. W. P. High-Selectivity Filter Banks for Spectral Analysis of Music Signals. *EURASIP Journal on Advances in Signal Processing*, Volume 2007:1-12, 2007.

[4] Fitch, J., Shabana, W. A Wavelet-based Pitch Detector for Musical Signals. Department of Mathematical Sciences, University of Bath, 1999.

[5] Fletcher, H., Munson, W. A., Loudness, its definition, measurement and calculation. *Journal of the Acoustical Society of America*, vol.5, 1933.

[6] Gera, Z. *Dallamfelismerés és kottázás valós időben.* Master Thesis, ELTE, 2004.

[7] Hacihabiboglu, H., Canagarajah, N. Instrument Recognition Based Wavelet Packet Trees in Audio Feature Extraction. *International Symposium on Musical Acoustics*, Digital Music Research Group, Department of Electrical and Electronic Engineering, University of Bristol, 2001.

[8] He, X., Scordilis, M. S. Psychoacoustic Music Analysis Based on the Discrete Wavelet Packet Transform. *Research Letters in Signal Processing*, Volume 2008:1-5, 2008.

[9] Iakovides, S. A., Iliadou, V. T., Bizeli, V. T., Kaprinis, S. G., Fountoulakis, K. N., Kaprinis, G. S. Psychophysiology and psychoacoustics of music: Perception of complex sound in normal subjects and psychiatric patients. *Annals of General Hospital Psychiatry*, 3:6:1-4, 2004.

[10] Jun, Z., Lei, G., Deyun, Z. A High-performance Psychoacoustics Approach to Speech Quality Evaluation. *Information Technology Journal*, 5(3): 485-488, 2006. Asian Network for Scientific Information

[11] Liu, K. J. R., Wu, A., Raghupathy, A., Chen, J. Algorithm-Based Low-Power and High-Performance Multimedia Signal Processing. IEEE, 1997.

[12] Mannell, R.H., The perceptual and auditory implications of parametric scaling in synthetic speech. Massachusetts Institute of Technology, Cambridge, Massachusetts, 1994.

[13] Mendel, A. Pitch in Western Music since 1500. A Re-Examination. *Acta Musicologica*, 1978.

[14] Meyer, Y. *Wavelets, Algorithms & Applications.* Society for Industrial and Applied Mathematics, Philadelphia, 1993.

[15] Obleser, J., Elbert, T., Eulitz, C. Attentional influences on functional mapping of speech sounds in human auditory cortex. *BMC Neuroscience*, 5:24:1-9, 2004.

[16] Okamoto, H., Kakigi, R., Gunji, A., Pantev, C. Asymmetric lateral inhibitory neural activity in the auditory system: a magnetoencephalographic study. *BMC Neuroscience*, 8:33:1-6, 2007.

[17] Poeppel, D. The Analysis of Speech in Different Temporal Integration Windows: Cerebral Laterization as Assymetric Sampling in Time. *Speech Communication* Volume 41, Issue 1, 2003.

[18] Brown, J. C., Puckette, M. S. An efficient algorithm for the calculation of a constant Q transform. *Journal of the Acoustical Society of America*, vol. 92, no. 5, 1992.

[19] Slaney, M., Lyon, R. F. A Perceptual Pitch Detector. *International Conference on Acoustics Speech and Signal Processing*, 1990.

[20] Todt, D. From birdsong to speech: a plea for comparative approaches. *Anais da Academia Brasileira de Ciencias*, 76(2):201-208, 2004.

Received 22nd August 2007

Acta Cybernetica **19** (2009) 177–190.

Complex Independent Process Analysis*

Zoltán Szabó† and András Lőrincz‡

Abstract

We present a general framework for the search of hidden independent processes in the complex domain. The task is to estimate the hidden independent multidimensional complex-valued components observing only the mixture of the processes driven by them. In our model (i) the hidden independent processes can be multidimensional, they may be subject to (ii) moving averaging, or may evolve in an autoregressive manner, or (iii) they can be non-stationary. These assumptions are covered by integrated autoregressive moving average processes and thus our task is to solve their complex extensions. We show how to reduce the undercomplete version of complex integrated autoregressive moving average processes to real independent subspace analysis that we can solve. Simulations illustrate the working of the algorithm.

1 Introduction

Our task is to find multidimensional independent components for complex variables. This task joins complex-valued neural networks [6] with independent component analysis (ICA) [4]. Although (i) complex-valued neural networks have several successful applications and (ii) there is a natural tendency to apply complex-valued computations for the analysis of biomedical signals (see, e.g., [2] and [3] for the analysis of EEG and fMRI data, respectively) the methodology of searching complex-valued independent components is barely treated in the literature. There are existing methods for the simplest ICA and blind source deconvolution tasks, but — to the best of our knowledge — there has been no study on non-i.i.d multidimensional hidden variables for the complex case. We provide the tools for this important problem family.

The paper is structured as follows: We treat the simplest complex-valued independent subspace analysis (complex ISA) problem and its solution in Section 2. In

*This research has been supported by the EC NEST 'PERCEPT' Grant FP6-043261. Opinions and errors in this manuscript are the author's responsibility, they do not necessarily reflect those of the EC or other project members.

†Department of Information Systems, Eötvös Loránd University, Pázmány Péter sétány 1/C, H-1117 Budapest, Hungary; E-mail: szzoli@cs.elte.hu

‡Corresponding author; Department of Information Systems, Eötvös Loránd University, Pázmány Péter sétány 1/C, H-1117 Budapest, Hungary; E-mail: andras.lorincz@elte.hu

the next section, the more general task, the complex-valued integrated autoregressive moving average independent subspace task is formulated. Section 4 contains our numerical illustrations. Conclusions are drawn in Section 5. The Appendix elaborates on the reduction technique: we show the series of transcriptions that reduce this task family to real independent subspace analysis.

2 The ISA Model

Below, in Section 2.1 we introduce the independent subspace analysis (ISA) problem. We show, how to reduce the complex-valued case to the real-valued one in Section 2.2.

2.1 The ISA Equations

We provide a joined formalism below for both the real and the complex-valued ISA models. To do so, let $\mathbb{K} \in \{\mathbb{R}, \mathbb{C}\}$ may stand for either real or for complex numbers and $\mathbb{K}^{D_1 \times D_2}$ denote the set of $D_1 \times D_2$ matrices over $\mathbb{K}$. The definition of the ISA task is as follows. We assume M pieces of hidden independent multidimensional random variables (*components*).[1] Only the linear mixture of these variables is available for observation. Formally,

$$\mathbf{x}(t) = \mathbf{A}\mathbf{e}(t), \tag{1}$$

where $\mathbf{e}(t) = \left[\mathbf{e}^1(t); \ldots; \mathbf{e}^M(t)\right] \in \mathbb{K}^{D_e}$ ($D_e = Md$) is a vector concatenated of the independent components $\mathbf{e}^m(t) \in \mathbb{R}^d$, where – for the sake of notational simplicity – we used identical dimension for each components. The dimensions of observation $\mathbf{x}$ and hidden source $\mathbf{e}$ are D_x and D_e, respectively. $\mathbf{A} \in \mathbb{K}^{D_x \times D_e}$ is the so-called *mixing matrix*. The goal of the ISA task is to estimate the original source $\mathbf{e}(t)$ from observations $\mathbf{x}(t)$. Our ISA assumptions are the followings:

1. For a given m, $\mathbf{e}^m(t)$ is i.i.d. in time t.
2. $I(\mathbf{e}^1, \ldots, \mathbf{e}^M) = 0$, where I stands for the mutual information of the arguments.
3. $\mathbf{A} \in \mathbb{K}^{D_x \times D_s}$ has full column rank, so it has a left inverse.

If $\mathbb{K} = \mathbb{C}$, then one can talk about complex-valued ISA (complex ISA). For the case of $\mathbb{K} = \mathbb{R}$, the ISA task is real-valued. The particular case of $d = 1$ gives rise to the ICA task.

[1] An excellent review can be found in [5] on complex random variables. Throughout this paper all complex variables are assumed to be full, i.e., they are not concentrated in any lower dimensional complex subspace.

2.2 Reduction of Complex-valued ISA to Real-valued ISA

In what follows, the complex ISA task is reduced to a real-valued one. To do so, consider the mappings

$$\varphi_v : \mathbb{C}^L \ni \mathbf{v} \mapsto \mathbf{v} \otimes \begin{bmatrix} \Re(\cdot) \\ \Im(\cdot) \end{bmatrix} \in \mathbb{R}^{2L}, \tag{2}$$

$$\varphi_M : \mathbb{C}^{L_1 \times L_2} \ni \mathbf{M} \mapsto \mathbf{M} \otimes \begin{bmatrix} \Re(\cdot) & -\Im(\cdot) \\ \Im(\cdot) & \Re(\cdot) \end{bmatrix} \in \mathbb{R}^{2L_1 \times 2L_2}, \tag{3}$$

where $\otimes$ is the Kronecker product, $\Re$ stands for the real part, $\Im$ for the imaginary part, subscript 'v' ('M') for vector (matrix). Known properties of mappings φ_v, φ_M are as follows [8]:

$$\det[\varphi_M(\mathbf{M})] = |\det(\mathbf{M})|^2 \qquad (\mathbf{M} \in \mathbb{C}^{L \times L}), \tag{4}$$

$$\varphi_M(\mathbf{M}_1\mathbf{M}_2) = \varphi_M(\mathbf{M}_1)\varphi_M(\mathbf{M}_2) \qquad (\mathbf{M}_1 \in \mathbb{C}^{L_1 \times L_2}, \mathbf{M}_2 \in \mathbb{C}^{L_2 \times L_3}), \tag{5}$$

$$\varphi_v(\mathbf{M}\mathbf{v}) = \varphi_M(\mathbf{M})\varphi_v(\mathbf{v}) \qquad (\mathbf{M} \in \mathbb{C}^{L_1 \times L_2}, \mathbf{v} \in \mathbb{C}^{L_2}), \tag{6}$$

$$\varphi_M(\mathbf{M}_1 + \mathbf{M}_2) = \varphi_M(\mathbf{M}_1) + \varphi_M(\mathbf{M}_2) \qquad (\mathbf{M}_1, \mathbf{M}_2 \in \mathbb{C}^{L_1 \times L_2}), \tag{7}$$

$$\varphi_M(c\mathbf{M}) = c\varphi_M(\mathbf{M}) \qquad (\mathbf{M} \in \mathbb{C}^{L_1 \times L_2}, c \in \mathbb{R}). \tag{8}$$

In words: (4) describes transformation of determinant, while (5), (6), (7) and (8) expresses preservation of operation for matrix-matrix multiplication, matrix-vector multiplication, matrix addition, real scalar-matrix multiplication, respectively.

Now, one may apply φ_v to the complex ISA equation ((1) with $\mathbb{K} = \mathbb{C}$) and use (6). The result is as follows:

$$\varphi_v(\mathbf{x}) = \varphi_M(\mathbf{A})\varphi_v(\mathbf{e}). \tag{9}$$

Given that (i) the independence of $\mathbf{e}^m \in \mathbb{C}^d$ is equivalent to that of $\varphi_v(\mathbf{e}^m) \in \mathbb{R}^{2d}$, and (ii) the existence of the left inverse of $\varphi_M(\mathbf{A})$ is inherited from $\mathbf{A}$ (see Eq. (5)), we end up with a real-valued ISA task with observation $\varphi_v(\mathbf{x})$ and M pieces of $2d$-dimensional hidden components $\varphi_v(\mathbf{e}^m)$.

3 Complex-valued Integrated Autoregressive Moving Average Independent Subspace Analysis

The solution of the complex-valued ISA task is important, because a series of transcriptions enables one to reduce much more general processes to it. We elaborate on this transcription series in the Appendix. Here, we provide the end result of this series, the model for complex-valued integrated autoregressive moving average independent subspace analysis. This will be the subject of our illustrations in the next section.

The complex-valued autoregressive moving average independent subspace task is this:

$$\mathbf{s}(t) = \sum_{i=1}^{p} \mathbf{P}_i \mathbf{s}(t-i) + \sum_{j=0}^{q} \mathbf{Q}_j \mathbf{e}(t-j), \tag{10}$$

$$\mathbf{x}(t) = \mathbf{A}\mathbf{s}(t). \tag{11}$$

These equations can be written in a more compact form by introducing the notations $\mathbf{P}[z] := \mathbf{I} - \sum_{i=1}^{p} \mathbf{P}_i z^i \in \mathbb{K}[z]^{D_s \times D_s}$ and $\mathbf{Q}[z] := \sum_{j=0}^{q} \mathbf{Q}_j z^j \in \mathbb{K}[z]^{D_s \times D_e}$. Here, polynomial matrices $\mathbf{P}[z]$ and $\mathbf{Q}[z]$ represent the autoregressive (AR) and the moving average (MA) parts, respectively. Now, we can simply write the complex-valued autoregressive moving average independent subspace task as this:

$$\mathbf{P}[z]\mathbf{s} = \mathbf{Q}[z]\mathbf{e}, \tag{12}$$

$$\mathbf{x} = \mathbf{A}\mathbf{s}. \tag{13}$$

Now, we provide the definition of the complex-valued *integrated* autoregressive moving average independent subspace task. This means that the difference process is complex-valued autoregressive moving average process. For the sake of notational transparency, let $\nabla^r[z] := (\mathbf{I} - \mathbf{I}z)^r$ denote the operator of the r^{th} order difference ($0 \leq r \in \mathbb{Z}$), where $\mathbf{I}$ is the identity matrix. Then, the general integrated task as is follows. We assume M pieces of hidden independent random variables (*components*). Only the linear mixture of $ARIMA(p, r, q)$ ($0 \leq p, r \in \mathbb{Z}$; $-1 \leq q \in \mathbb{Z}$) processes driven by these hidden components is available for observation. Formally,

$$\mathbf{P}[z]\nabla^r[z]\mathbf{s} = \mathbf{Q}[z]\mathbf{e}, \tag{14}$$

$$\mathbf{x} = \mathbf{A}\mathbf{s}, \tag{15}$$

where $\mathbf{e}(t) = \left[\mathbf{e}^1(t); \ldots; \mathbf{e}^M(t)\right] \in \mathbb{K}^{D_e}$ ($D_e = Md$) is a vector concatenated of the independent components $\mathbf{e}^m(t) \in \mathbb{R}^d$. Observation $\mathbf{x} \in \mathbb{K}^{D_x}$, hidden source $\mathbf{s} \in \mathbb{K}^{D_s}$, mixing matrix $\mathbf{A} \in \mathbb{K}^{D_x \times D_s}$, polynomial matrices $\mathbf{P}[z] := \mathbf{I} - \sum_{i=1}^{p} \mathbf{P}_i z^i \in \mathbb{K}[z]^{D_s \times D_s}$ and $\mathbf{Q}[z] := \sum_{j=0}^{q} \mathbf{Q}_j z^j \in \mathbb{K}[z]^{D_s \times D_e}$. The task is to estimate the original source $\mathbf{e}(t)$ from observations $\mathbf{x}(t)$.

The conditions, when we can reduce the complex-valued *integrated* autoregressive moving average independent subspace task to an ISA task are as follows:

1. For a given m, $\mathbf{e}^m(t)$ is i.i.d. in time t.
2. $I(\mathbf{e}^1, \ldots, \mathbf{e}^M) = 0$.
3. $\mathbf{A} \in \mathbb{C}^{D_x \times D_s}$ has full column rank.
4. Polynomial matrix $\mathbf{P}[z]$ is stable ($det(\mathbf{P}[z])$ has no roots within the closed unit circle).
5. The task is undercomplete: $D_x > D_e$.

The case of $r = 0$ corresponds to the complex-valued autoregressive moving average independent subspace task. Details of the series of transcriptions can be found in the Appendix. The interested reader may find further details and a number of references about multidimensional independent component analysis in [13].

4 Illustrations

The complex-valued integrated autoregressive moving average independent subspace analysis problem can be reduced to a real ISA task as it is detailed in Appendix B. Here we illustrate the performance of the algorithm based on those reductions. To evaluate the solutions we use a performance measure given in Section 4.1. Our test database is described in Section 4.2. Numerical results are summarized in Section 4.3.

4.1 Performance Index

Using the reduction principle of Section B, in the ideal case, the product of matrix $\varphi_M(\mathbf{A})\varphi_M(\mathbf{Q}_0)$ and the matrices provided by PCA (principal component analysis), ISA, i.e., $\mathbf{G} := (\hat{\mathbf{W}}_{\mathrm{ISA}}\hat{\mathbf{W}}_{\mathrm{PCA}})\varphi_M(\mathbf{A})\varphi_M(\mathbf{Q}_0) \in \mathbb{R}^{2D_e \times 2D_e}$ is a block-permutation matrix made of $2d \times 2d$ blocks. This block-permutation structure can be measured by the normalized version of the Amari-error [1] adapted to the ISA task [19]. Let us decompose matrix $\mathbf{G} \in \mathbb{R}^{2D_e \times 2D_e}$ into blocks of size $2d \times 2d$: $\mathbf{G} = [\mathbf{G}_{ij}]_{i,j=1,\ldots,M}$. Let g_{ij} denote the sum of the absolute values of matrix $\mathbf{G}_{ij} \in \mathbb{R}^{2d \times 2d}$. Now, the following term [15]

$$r(\mathbf{G}) := \frac{1}{2M(M-1)} \left[\sum_{i=1}^{M} \left(\frac{\sum_{j=1}^{M} g_{ij}}{\max_j g_{ij}} - 1 \right) + \sum_{j=1}^{M} \left(\frac{\sum_{i=1}^{M} g_{ij}}{\max_i g_{ij}} - 1 \right) \right] \tag{16}$$

denotes the Amari-index that takes values in [0,1]: for an ideal block-permutation matrix $\mathbf{G}$ it takes 0; for the worst case it takes 1.

4.2 Test Database

We created a database for the illustration, which is scalable in dimension d. The hidden sources $\mathbf{e}^m$ were defined by geometrical forms in $\mathbb{C}^d$. Using that $\varphi_v : \mathbb{C}^d \to \mathbb{R}^{2d}$ is a bijection, variables $\mathbf{e}^m$ were created in $\mathbb{R}^{2d}$. Namely, we used geometrical forms in $\mathbb{R}^{2d}$, applied uniform sampling on these and the φ_v^{-1} derived image of the samples $\mathbb{R}^{2d}$ was taken as $\mathbf{e}^m \in \mathbb{C}^d$. Geometrical forms were chosen as follows. We used: (i) the surface of the unit ball, (ii) the straight lines that connect the opposing corners of the unit cube, (iii) the broken line between $2d + 1$ points $\mathbf{0} \to \mathbf{e}_1 \to \mathbf{e}_1 + \mathbf{e}_2 \to \ldots \to \mathbf{e}_1 + \ldots + \mathbf{e}_{2d}$ (where $\mathbf{e}_i$ is the i canonical basis vector in $\mathbb{R}^{2d}$, i.e., all of its coordinates are zero except the i, which is 1), and (iv) the skeleton of the unit square. Thus in our numerical studies the number of components M was equal to 4. For illustration, see Fig 1.

(a) (b) (c) (d)

Figure 1: Illustration of our test database. Hidden components $\mathbf{e}^m \in \mathbb{C}^d$ are defined as variables uniformly distributed on geometrical forms, shown here. For this, bijection $\varphi_v : \mathbb{C}^d \to \mathbb{R}^{2d}$ was used. The figure serves illustrative purposes only, because $2d$ is even.

4.3 Simulations

We present our simulation results here. We focus on 2 distinct issues:

1. How does the estimation error scale with the number of samples? Sample number T ranged between $2,000 \leq T \leq 30,000$ and the orders of the AR and MA processes were kept low: $p = 1$, $q = 1$ (precisely, MA order: $q + 1 = 2$).

2. We assumed that polynomial matrix $\mathbf{P}[z]$ of Eq. (14) is stable. In the case of $r = 0$ this means that process $\mathbf{s}$ is stationary. For $r > 1$ the model describes non-stationary processes. It is expected that if the roots of $\mathbf{P}[z]$ are close to the unit circle then our estimation will deteriorate. We investigated this by generating polynomial matrix $\mathbf{P}[z]$ as follows:

$$\mathbf{P}[z] = \prod_{i=1}^{p} (\mathbf{I} - \lambda \mathbf{U}_i z) \quad (|\lambda| < 1, \lambda \in \mathbb{R}) \tag{17}$$

Matrices $\mathbf{U}_i \in \mathbb{C}^{D_s \times D_s}$ were random unitary matrices and the $\lambda \to 1$ limit was studied. Now, sample number was set to $T = 20,000$. For the 'small task' ($p = 1$, $q = 1$) we could not see relevant performance drops even for $\lambda = 0.99$, therefore we increased parameters p and q to 5 and 10, respectively.

In our simulations: (i) the measure of undercompleteness was 2 ($D_x = D_s = 2D_e$), (ii) the Amari-index was used to measure the precision of our method. For all values of parameters (T, p, r, q), the average performances upon 20 random initializations of $\mathbf{e}, \mathbf{Q}[z], \mathbf{P}[z]$ and $\mathbf{A}$ were taken. In economic computations, the value of r is typically ≤ 2, we investigated values between $1 \leq r \leq 3$. The coordinates of matrices $\mathbf{Q}_j$ in the MA part (see Eq. (14)) were chosen independently and uniformly from the $\{\mathbf{v} = v_1 + iv_2 \in \mathbb{C} : -\frac{1}{2} \leq v_1, v_2 \leq \frac{1}{2}\}$ complex unit square. The mixing matrix $\mathbf{A}$ (see, Eq. (15)) was drawn randomly from the unitary group. Polynomial matrix $\mathbf{P}[z]$ was generated according to Eq. (17). The choice of λ is detailed later. The order of the AR estimation (see Fig. 4) was constrained from above as follows $deg(\hat{\mathbf{W}}_{\mathrm{AR}}[z]) \leq 2(q + 1) + p$ (i.e., two times the MA length + the AR length). We used the technique of [9] with the Schwarz's Bayesian Criterion to determine

the optimal order of the AR process. We applied the method of [14] to solve the associated ISA task.

In our first test the order of AR (p) and the order of MA processes (q) were set to the minimal meaningful values, 1. We investigated the estimation error as a function of the sample number. Parameter r of the process was set to $r = 1, 2$ and 3 in the different computations. Sample number varied as $T = 2, 5, 10, 20, 30 \cdot 10^3$. Scaling properties of the algorithm were studied by changing the value of the dimension of the components d between 1 and 15. The value of λ was 0.9 (see, Eq. (17)). Our results are summarized in Fig. 2(c), with an illustrative example given in Fig. 2(a)-(d).[2] According to Fig. 2(c), our method could recover the hidden components with high precision. The Amari-index $r(T)$ follows power law $r(T) \propto T^{-c}$ ($c > 0$). The power law is manifested by straight lines on loglog scales. The slope of the lines are about the same for different d values. The actual values of the Amari-index can be found in Table 1 for sample number $T = 30,000$.

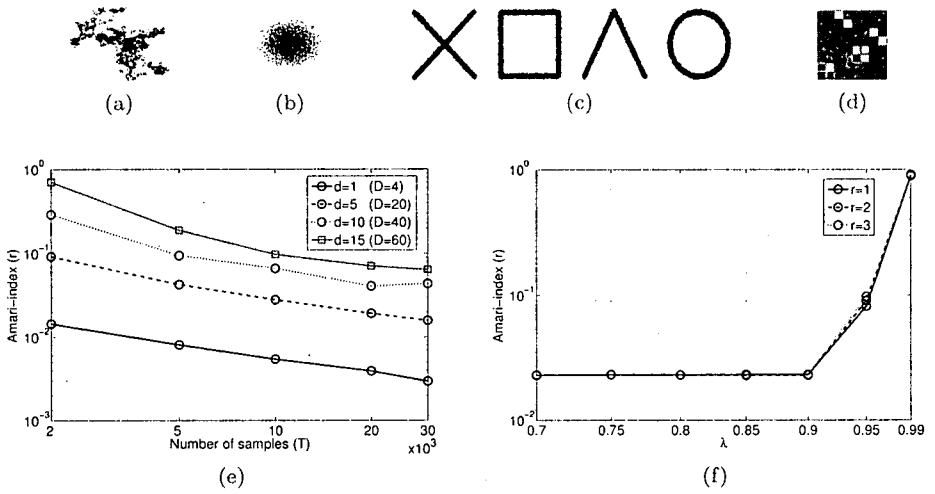


Figure 2: Illustration of our method. (a)-(d): AR order $p = 1$, MA order $q = 1$, order of integration $r = 1$, sample number $T = 30,000$. (a)-(b): typical 2D projection of the observed mixed $\mathbf{x}$ signal, and its r^{th}-order difference. (c): estimated components $[\varphi_v(\mathrm{e}^m)]$. (d): Hinton-diagram of $\mathbf{G}$, ideally block-permutation matrix with 2×2 blocks. (e): average Amari-index as a function of the sample size on loglog scale for different dimensional (d) components; $\lambda = 0.9$, $p = 1$, $q = 1$, $r = 1$ ($r \leq 3$). For $T = 30,000$, the exact errors are shown in Table 1. (f): Estimation error on log scale as a function of the magnitude of the roots of polynomial matrix $\mathbf{P}[z]$. (If $\lambda = 1$ then the roots are on the unit circle.) Parameters: $r = 1, 2$ and 3; AR order: $p = 5$; MA order: $q = 10$.

[2] The $r = 1$ case is illustrated, results are similar in the studied $r \leq 3$ domain.

In our other test we investigated what happens if the roots of polynomial matrix $\mathbf{P}[z]$ move towards the unit circle from the outside. In these simulations, parameter λ of Eq. (17) was varied. Our question was the following: How does our method behave when λ is close to 1? The sample number was set to $T = 20,000$ and simultaneously the AR order p, and MA order q were increased to 5 and 10, respectively. Dimension d of components $\mathbf{e}^m$ was 5. Parameter r took values on $1, 2$ and 3. Results are shown in Fig. 2(f). According to this figure, there is a sudden change in the performance at around $\lambda = 0.9 - 0.95$. Estimations for parameters $r = 1, 2$ and 3 have about the same errors. We note that for $p = 1$ and $q = 1$ we did not experience any degradation of performance up to $\lambda = 0.99$.

$d = 1$	$d = 5$	$d = 10$	$d = 15$
0.29% (±0.05)	1.59% (±0.05)	4.36% (±2.61)	6.40% (±3.10)

Table 1: Amari-index as a function of the dimension of the components d: average ± std. Sample size: $T = 30,000$. For other sample numbers, see Fig. 2(e).

5 Conclusions

We have given a general framework for the search of hidden independent components in the complex domain. This integrated autoregressive moving average subspace problem formulation can cover several distinct assumptions. The hidden processes (i) may be multidimensional, (ii) can be autoregressive or moving average processes, and (iii) may be non-stationary processes, too. We have shown that the undercomplete version of this task can be reduced to real-valued ISA problem. We investigated the efficiency of our method by means of numerical simulations. We experienced that (i) the estimation error decreases and follows a power law as a function of the sample number and (ii) the estimation is robust if the AR term is stable.

References

[1] Amari, Shun-ichi, Cichocki, Andrzej, and Yang, Howard H. A new learning algorithm for blind signal separation. *Advances in Neural Information Processing Systems*, 8:757–763, 1996.

[2] Anemüller, Jörn, Sejnowski, Terrence J., and Makeig, Scott. Complex ICA of frequency-domain electroencephalographic data. *Neural Networks*, 16:1311–1323, 2003.

[3] Calhoun, Vince D. and Adali, Tülay. Complex infomax: Convergence and approximation of infomax with complex nonlinearities. *VLSI Signal Processing Systems for Signal, Image, and Video Technology*, 44(1/2):173–190, 2006.

[4] Cichocki, Andrzej and Amari, Shun-ichi. *Adaptive blind signal and image processing*. John Wiley & Sons, 2002.

[5] Eriksson, Jan. Complex random vectors and ICA models: Identifiability, uniqueness and separability. *IEEE Transactions on Information Theory*, 52(3), 2006.

[6] Hirose, Akira. *Complex-Valued Neural Networks: Theories and Applications*, volume 5 of *Series on Innovative Intelligence*. World Scientific Publishing Co. Pte. Ltd., 2004.

[7] Hyvärinen, Aapo. Independent component analysis for time-dependent stochastic processes. In *Proceedings of ICANN*, pages 541–546, Berlin, 1998. Springer-Verlag.

[8] Krishnaiah, P. and Lin, Jugan. Complex elliptically symmetric distributions. *Communications in Statistics*, 15(12):3693–3718, 1986.

[9] Neumaier, Arnold and Schneider, Tapio. Estimation of parameters and eigenmodes of multivariate autoregressive models. *ACM Transactions on Mathematical Software*, 27(1):27–57, 2001.

[10] Póczos, Barnabás, Szabó, Zoltán, Kiszlinger, Melinda, and Lőrincz, András. Independent process analysis without a priori dimensional information. In *Proceedings of ICA*, volume 4666 of *LNCS*, pages 252–259. Springer-Verlag, 2007.

[11] Póczos, Barnabás, Takács, Bálint, and Lőrincz, András. Independent subspace analysis on innovations. In *Proceedings of ECML*, pages 698–706. Springer-Verlag, 2005.

[12] Rajagopal, Ravikiran and Potter, Lee C. Multivariate MIMO FIR inverses. *IEEE Transactions on Image Processing*, 12:458 – 465, 2003.

[13] Szabó, Zoltán. *Separation Principles in Independent Process Analysis*. PhD thesis, Eötvös Loránd University, Budapest, 2008. (submitted; available at `http://nipg.inf.elte.hu/index.php?option=com_remository&Itemid=27&func=fileinfo&id=153`).

[14] Szabó, Zoltán and Lőrincz, András. Real and complex independent subspace analysis by generalized variance. In *Proceedings of ICARN*, pages 85–88, Liverpool, U.K., 2006. (Available at `http://arxiv.org/abs/math.ST/0610438`).

[15] Szabó, Zoltán, Póczos, Barnabás, and Lőrincz, András. Cross-entropy optimization for independent process analysis. In *Proceedings of ICA*, volume 3889 of *LNCS*, pages 909–916. Springer, 2006.

[16] Szabó, Zoltán, Póczos, Barnabás, and Lőrincz, András. Undercomplete blind subspace deconvolution. *Journal of Machine Learning Research*, 8:1063–1095, 2007.

[17] Szabó, Zoltán, Póczos, Barnabás, and Lőrincz, András. Undercomplete blind subspace deconvolution via linear prediction. In *Proceedings of ECML*, volume 4701 of *LNAI*, pages 740–747. Springer-Verlag, 2007.

[18] Theis, Fabian J. Uniqueness of complex and multidimensional independent component analysis. *Signal Processing*, 84(5):951–956, 2004.

[19] Theis, Fabian J. Multidimensional independent component analysis using characteristic functions. In *Proceedings of EUSIPCO*, Antalya, Turkey, 2005.

Appendix

In this Appendix we elaborate on the details of the general $\mathbb{K}$-ARIMA-IPA model. Section A: we describe special cases, going step-by-step to more general process models. In Section B we reduce the complex-valued ARIMA-IPA task to the real-valued case. This reduction is analogous to the main lines of Section 2.2.

A The $\mathbb{K}$-ARIMA-IPA Equations

We defined the ISA task in Section 2.1. In case of ISA, one assumes that the hidden sources are independent and identically distributed (i.i.d.) in time. Temporal independence is, however, a gross oversimplification of sources. Temporal dependencies can be diminished, e.g., by an

- autoregressive (AR) assumption for the hidden sources. This is the AR independent process analysis (AR-IPA) task [7, 11]:

$$\mathbf{s}(t) = \sum_{i=1}^{p} \mathbf{P}_i \mathbf{s}(t-i) + \mathbf{Q}_0 \mathbf{e}(t), \tag{18}$$

$$\mathbf{x}(t) = \mathbf{A}\mathbf{s}(t). \tag{19}$$

  Here, we assume the i.i.d. property for driving noise $\mathbf{e}(t)$, but not for hidden source $\mathbf{s}(t)$. The state equation ((18)) and the observation ((19)) can be written compactly using the polynomial matrix formalism: let z stand for the time-shift operation, that is $(z\mathbf{v})(t) := \mathbf{v}(t-1)$ and polynomials of $D_1 \times D_2$ matrices are denoted as $\mathbb{K}[z]^{D_1 \times D_2} := \{\mathbf{F}[z] = \sum_{n=0}^{N} \mathbf{F}_n z^n, \mathbf{F}_n \in \mathbb{K}^{D_1 \times D_2}\}$. Then, Eqs. (18)-(19) take the forms:

$$\mathbf{P}[z]\mathbf{s} = \mathbf{Q}_0 \mathbf{e}, \tag{20}$$

$$\mathbf{x} = \mathbf{A}\mathbf{s}, \tag{21}$$

  where $\mathbf{P}[z] := \mathbf{I} - \sum_{i=1}^{p} \mathbf{P}_i z^i \in \mathbb{K}[z]^{D_s \times D_s}$ represents the AR part of order p. For $p = 0$, the $\mathbb{K}$-valued ISA ($\mathbb{K}$-ISA) task is recovered.

- moving average (MA) assumption. The observation in the $\mathbb{K}$-MA-IPA task (which could also be called $\mathbb{K}$-blind subspace deconvolution (BSSD) [16, 17]) task) is as follows:

$$\mathbf{x}(t) = \sum_{j=0}^{q} \mathbf{Q}_j \mathbf{e}(t-j). \tag{22}$$

In polynomial matrix form

$$\mathbf{x} = \mathbf{Q}[z]\mathbf{e}, \tag{23}$$

where $\mathbf{Q}[z] := \sum_{j=0}^{q} \mathbf{Q}_j z^j \in \mathbb{K}[z]^{D_s \times D_e}$ represents the MA part of order q. Here:

  - for $q = 0$ the $\mathbb{K}$-ISA task appears.
  - If $d = 1$ holds, then we end up with the $\mathbb{K}$-BSD ($\mathbb{K}$-blind source deconvolution) problem.

Combining the AR and the MA assumptions the $\mathbb{K}$-ARMA-IPA task emerges:

$$\mathbf{s}(t) = \sum_{i=1}^{p} \mathbf{P}_i \mathbf{s}(t-i) + \sum_{j=0}^{q} \mathbf{Q}_j \mathbf{e}(t-j), \tag{24}$$

$$\mathbf{x}(t) = \mathbf{A}\mathbf{s}(t), \tag{25}$$

which can be written compactly as

$$\mathbf{P}[z]\mathbf{s} = \mathbf{Q}[z]\mathbf{e}, \tag{26}$$

$$\mathbf{x} = \mathbf{A}\mathbf{s}, \tag{27}$$

where $\mathbf{P}[z] := \mathbf{I} - \sum_{i=1}^{p} \mathbf{P}_i z^i \in \mathbb{K}[z]^{D_s \times D_s}$ and $\mathbf{Q}[z] := \sum_{j=0}^{q} \mathbf{Q}_j z^j \in \mathbb{K}[z]^{D_s \times D_e}$. For the general ARMA process the condition is that polynomial matrix $\mathbf{P}[z]$ is *stable*, that is $\det(\mathbf{P}[z]) \neq 0$, for all $z \in \mathbb{C}, |z| \leq 1$. We note that the stability of $\mathbf{P}[z]$ implies the stationarity of ARMA process $\mathbf{s}$.

Using temporal differences, we enter the domain of non-stationary processes. In such case the ARMA property is assumed for the first order difference process $\mathbf{s}(t) - \mathbf{s}(t-1)$, or similarly for higher order difference processes. For the general order r, let $\nabla^r[z] := (\mathbf{I} - \mathbf{I}z)^r$ denote the operator of the r^{th} order difference ($0 \leq r \in \mathbb{Z}$), where $\mathbf{I}$ is the identity matrix. Then, the definition of the $\mathbb{K}$-ARIMA-IPA task as is follows. We assume M pieces of hidden independent random variables (*components*). Only the linear mixture of $ARIMA(p, r, q)$ ($0 \leq p, r \in \mathbb{Z}$; $-1 \leq q \in \mathbb{Z}$) processes driven by these hidden components is available for observation. Formally,

$$\mathbf{P}[z]\nabla^r[z]\mathbf{s} = \mathbf{Q}[z]\mathbf{e}, \tag{28}$$

$$\mathbf{x} = \mathbf{A}\mathbf{s}, \tag{29}$$

where $\mathbf{e}(t) = \left[\mathbf{e}^1(t); \ldots; \mathbf{e}^M(t)\right] \in \mathbb{K}^{D_e}$ ($D_e = Md$) is a vector concatenated of the independent components $\mathbf{e}^m(t) \in \mathbb{R}^d$. Observation $\mathbf{x} \in \mathbb{K}^{D_x}$, hidden source

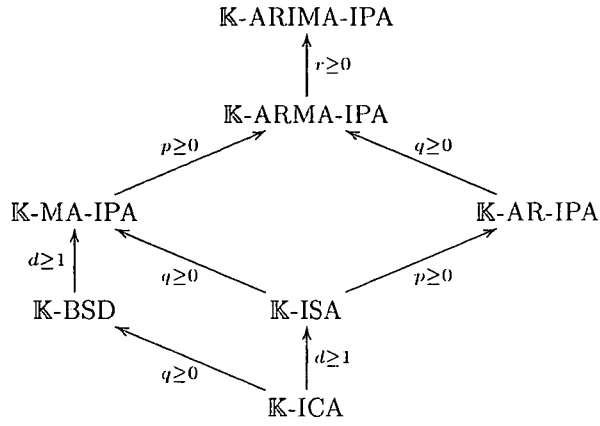


Figure 3: The $\mathbb{K}$-ARIMA-IPA model. Arrows show the direction of generalization. The labels of the arrows explain the method of the generalization. For example: '$\mathbb{K}$-ICA $\xrightarrow{d\geq 1}$ $\mathbb{K}$-ISA' means that the $\mathbb{K}$-ISA task is the generalization of the $\mathbb{K}$-ICA task such that the hidden independent sources may be multidimensional, i.e., $d \geq 1$.

$\mathbf{s} \in \mathbb{K}^{D_s}$, mixing matrix $\mathbf{A} \in \mathbb{K}^{D_x \times D_s}$, polynomial matrices $\mathbf{P}[z] := \mathbf{I} - \sum_{i=1}^{p} \mathbf{P}_i z^i \in \mathbb{K}[z]^{D_s \times D_s}$ and $\mathbf{Q}[z] := \sum_{j=0}^{q} \mathbf{Q}_j z^j \in \mathbb{K}[z]^{D_s \times D_e}$. The goal of the $\mathbb{K}$-ARIMA-IPA task is to estimate the original source $\mathbf{e}(t)$ from observations $\mathbf{x}(t)$.

Our $\mathbb{K}$-ARIMA-IPA assumptions are listed below:

1. For a given m, $\mathbf{e}^m(t)$ is i.i.d. in time t.
2. $I(\mathbf{e}^1, \ldots, \mathbf{e}^M) = 0$.
3. $\mathbf{A} \in \mathbb{K}^{D_x \times D_s}$ has full column rank.
4. Polynomial matrix $\mathbf{P}[z]$ is stable.

The $\mathbb{K}$-ARMA-IPA task corresponds to the $r = 0$ case.

The relations amongst the different tasks are summarized in Fig. 3.

B Decomposition of the $\mathbb{C}$-uARIMA-IPA Model

Here, we reduce the $\mathbb{C}$-ARIMA-IPA task to $\mathbb{R}$-ISA for the *undercomplete* case ($D_x > D_e$; $\mathbb{C}$-uARIMA-IPA; letter 'u' is to show the restriction for the undercomplete case). The reduction takes two steps:

1. In Section B.1, the $\mathbb{C}$-uARIMA-IPA task is reduced to the $\mathbb{R}$-uARIMA-IPA task.

2. The $\mathbb{R}$-uARIMA-IPA task can be solved following the route suggested in [10], because it can be reduced to the $\mathbb{R}$-ISA task. The undercomplete assumption is used in the second step only. For the sake of completeness, we also provide a description of the second step (Section B.2).

In addition to the conditions of the ARIMA-IPA task, we assume that $\mathbf{Q}[z]$ has left inverse. In other words, there exists a polynomial matrix $\mathbf{W}[z] \in \mathbb{R}[z]^{D_e \times D_s}$ such that $\mathbf{W}[z]\mathbf{Q}[z] = \mathbf{I}_{D_e}$ (thus $D_s > D_e$)[3].

B.1 Reducing the Complex ARIMA-IPA Task to Real Variables

Here we reduce the tasks of Fig. 3, which have complex variables to real variables. In particular, we reduce the $\mathbb{C}$-uARIMA-IPA problem to the $\mathbb{R}$-uARIMA-IPA task.

One may apply φ_v to the (28)-(29) $\mathbb{C}$-ARIMA-IPA equations (with $\mathbb{K} = \mathbb{C}$) and use (6)-(8). The result is as follows:

$$\varphi_M(\mathbf{P}[z])\nabla^r[z]\varphi_v(\mathbf{s}) = \varphi_M(\mathbf{Q}[z])\varphi_v(\mathbf{e}), \tag{30}$$

$$\varphi_v(\mathbf{x}) = \varphi_M(\mathbf{A})\varphi_v(\mathbf{s}). \tag{31}$$

Given that (i) the independence of $\mathbf{e}^m \in \mathbb{C}^d$ is equivalent to that of $\varphi_v(\mathbf{e}^m) \in \mathbb{R}^{2d}$, and (ii) the stability of $\varphi_M(\mathbf{P}[z])$ and the existence of the left inverse of $\varphi_M(\mathbf{Q}[z])$ are inherited from $\mathbf{P}[z]$ and $\mathbf{Q}[z]$, respectively (see Eqs. (4) and (5)), we end up with an $\mathbb{R}$-ARIMA-IPA task with (p, r, q) parameters and M pieces of $2d$-dimensional hidden components $\varphi_v(\mathbf{e}^m)$.

B.2 Reduction of $\mathbb{R}$-uARIMA-IPA to $\mathbb{R}$-ISA

We ended up with a $\mathbb{R}$-uARIMA-IPA task in Section B.1. This task can be reduced to a $\mathbb{R}$-ISA task as it has been shown in [10]. The reduction requires two steps: (i) temporal differencing and (ii) linear prediction. These steps are formalized by the following lemmas:

Lemma 1. *Differentiating the observation* $\mathbf{x}$ *of an* $\mathbb{R}$*-(u)ARIMA-IPA task in* r^{th} *order one obtains an* $\mathbb{R}$*-(u)ARMA-IPA task:*

$$\mathbf{P}[z]\left(\nabla^r[z]\mathbf{s}\right) = \mathbf{Q}[z]\mathbf{e}, \tag{32}$$

$$\nabla^r[z]\mathbf{x} = \mathbf{A}\left(\nabla^r[z]\mathbf{s}\right), \tag{33}$$

(where, the relation $z\mathbf{x} = \mathbf{A}(z\mathbf{s})$ *has been used).*

We note that polynomial matrix $\varphi_M(\mathbf{Q}[z])$ derived from the $\mathbb{C}$-uARIMA-IPA task has a left inverse (see Section B.1). Thus, we can apply the above quoted linear prediction based result:

[3] One can show for $D_s > D_e$ that under mild conditions $\mathbf{Q}[z]$ has left inverse with probability 1 [12]; e.g., when the matrix $[\mathbf{Q}_0, \ldots, \mathbf{Q}_q]$ is drawn from a continuous distribution.

$$\mathbb{C}\text{-uARIMA-IPA} \xrightarrow[\varphi_v,\varphi_M]{} \mathbb{R}\text{-uARIMA-IPA} \xrightarrow[\nabla^r[z]]{} \mathbb{R}\text{-uARMA-IPA} \xrightarrow[\hat{\mathbf{W}}_{\mathrm{AR}}[z],\hat{\mathbf{W}}_{\mathrm{PCA}}]{} \mathbb{R}\text{-ISA}$$

Figure 4: Reduction of $\mathbb{C}$-uARIMA-IPA to $\mathbb{R}$-ISA. Prefix 'u': undercomplete case. Double arrows: transformations of the reduction steps. Estimated $\mathbb{R}$-ISA separation matrix: $\hat{\mathbf{W}}_{\mathrm{ISA}}$. $\hat{\mathbf{W}}_{\mathbb{R}-\mathrm{ARIMA}}[z] = \hat{\mathbf{W}}_{\mathrm{ISA}}\hat{\mathbf{W}}_{\mathrm{PCA}}\hat{\mathbf{W}}_{\mathrm{AR}}[z]\nabla^r[z]$. Estimated source: $\hat{\mathbf{W}}_{\mathbb{R}-\mathrm{ARIMA}}[z]\varphi_v(\mathbf{x})$, or after transforming back to the complex space $\hat{\mathbf{e}} = \varphi_v^{-1}[\hat{\mathbf{W}}_{\mathbb{R}-\mathrm{ARIMA}}[z]\varphi_v(\mathbf{x})]$.

Lemma 2. *In the $\mathbb{R}$-uARMA-IPA task, observation process $\mathbf{x}(t)$ is autoregressive and its innovation $\tilde{\mathbf{x}}(t) := \mathbf{x}(t) - E[\mathbf{x}(t)|\mathbf{x}(t-1), \mathbf{x}(t-2), \ldots]$ is $\mathbf{A}\mathbf{Q}_0\mathbf{e}(t)$, where $E[\cdot|\cdot]$ denotes the conditional expectation value. Consequently, there is a polynomial matrix $\mathbf{W}_{\mathrm{AR}}[z] \in \mathbb{R}[z]^{D_x \times D_x}$ such that $\mathbf{W}_{\mathrm{AR}}[z]\mathbf{x} = \mathbf{A}\mathbf{Q}_0\mathbf{e}$.*

Thus, AR fit of $\nabla^r[z]\,(\varphi_v[\mathbf{x}(t)])$ can be used for the estimation of $\varphi_M(\mathbf{A}\mathbf{Q}_0)\varphi_v[\mathbf{e}(t)]$. This innovation corresponds to the observation of an undercomplete $\mathbb{R}$-ISA model ($D_x > D_e$), which can be reduced to a complete $\mathbb{R}$-ISA ($D_x = D_e$) using principal component analysis (PCA). Finally, the solution can be finished by any $\mathbb{R}$-ISA procedure. The steps of our algorithm are summarized in Fig. 4.

The reduction procedure implies that the derived hidden components $\varphi_v(\mathbf{e}^m)$ can be recovered only up to the ambiguities of the $\mathbb{R}$-ISA task [18]: components of (identical dimensions) can be recovered only up to permutations. Within each subspaces, unambiguity is warranted only up to linear transformations that can be reduced to orthogonal transformations provided that both the hidden source ($\mathbf{e}$) and the observation are white; their expectation values are $\mathbf{0}$ and the covariance matrices are identity matrices. These conditions make no loss to the generality of our solution. Notice that the unitary property of matrix $\mathbf{M}$ is equivalent to the orthogonality of matrix $\varphi_M(\mathbf{M})$ [8]. Thus, apart from a permutation of the components, we can reproduce components $\mathbf{e}^m$ only up to an unitary transformation.

Received 18th July 2007

Acta Cybernetica **19** (2009) 191–216.

Plagiarism Detection in Source Programs Using Structural Similarities

Gergely Lukácsy* and Péter Szeredi*

Abstract

The paper presents a plagiarism detection framework the goal of which is to determine whether two programs are similar to each other, and if so, to what extent.

The issue of plagiarism detection has been considered earlier for written material, such as student essays. For these, text-based algorithms have been published. We argue that in case of program code comparison, structure based techniques may be much more suitable. The main idea is to transform the source code into mathematical objects, use appropriate reduction and comparison methods on these, and interpret the results appropriately.

We have designed a generic program structure comparison framework and implemented it for the Prolog and SML programming languages. We have been using the implementation at BUTE to successfully detect plagiarism in homework assignments for years.

Keywords: plagiarism, program source, graph similarity

1 Introduction and motivation

Comparison of essays and other written materials has been in focus in recent years [27]. Detecting plagiarism in written materials is an issue in education as well as in law procedures. World wide public polls show that two-thirds of university students have used other people's ideas in an impermissible way at least once during their studies. Law disputes include the SCO-IBM debate over the allegedly unauthorised use of portions of the AIX operating system in Linux.

Regrettably, several sites on the Internet provide free or low cost, quick and efficient access to written materials of many types. Unbelievably, sites such as `CheatHouse`[1] or `SchoolSucks`[2] proudly provide tons of essays, dissertations, reports, etc. for students looking for an easy way to have their assignment of some sort fulfilled. We do agree that it is a good idea to get acquainted with the area one

*Budapest University of Technology and Economics (BUTE), Department of Computer Science and Information Theory, 1117 Budapest, Magyar tudósok körútja 2., Hungary, Phone: +36 1 463-2585 Fax: +36 1 463-3157, E-mail: {lukacsy,szeredi}@cs.bme.hu

[1] `http://www.cheathouse.com`

[2] `http://www.schoolsucks.com`

is interested in by reading similar materials. However inspiring someone to cheat is a different issue.

In case of programming assignments, it is important to detect the duplication of programs or parts of these. Students attending the course "Declarative Programming" at BUTE are expected to hand in a major programming assignment at the end of the semester. This means mass amount of program sources year by year.

Checking these programs by hand seems to be beyond possibility. Having n programs we should check $\frac{n*(n-1)}{2}$ pairs to have all the cases covered. Notice, that we really should check all of the pairs, because the relation "P1 is similar to P2", where P1 and P2 are programs, is not transitive. This practically means that even if we know that source A is similar to source B and source B to source C we cannot draw any direct conclusion about the similarity degree of sources A and C.

Luckily, in our particular case several assignments can be excluded from the whole set. For example, we do not care whether two bad solutions are similar or not (a solution is bad if it does not solve a certain percentage of the given test cases). However we still have $O(n^2)$ pairs to test manually, where n is often greater than 100.

Our aim was to develop methods and tools to assess the similarity of programs in order to narrow down the need for manual testing to an acceptable amount. We have defined the notion of a *similarity degree* which reflects how much two programs match. For the methods to be generic and flexible enough we have developed a *multi phase comparison framework.*

The actual comparison is performed between mathematical entities where the meaning of similarity can be formally specified. These entities are generated from the programs to be compared. The procedure may vary for different programming languages, so separate front-end modules should be developed for each language. Naturally, the mathematical entities must be generic and powerful enough to be applicable to different languages. We have chosen directed, labelled graphs for this purpose. Now, the comparison of source programs is actually reduced to calculating the similarity measure of graphs. Notice, that this way it is also possible to determine the similarity degree of two programs written in different languages.

The framework is customisable, so that it remains usable under varying circumstances. For instance, in case of shorter programs a different similarity threshold may be more appropriate than in the case of bigger ones. Moreover, we found that applying certain well selected simplifying graph transformations, called *reductions*, has favourable effects on the efficiency of the approach. Such reductions include removing specific nodes and edges and thus creating higher level, more abstract views of the programs.

The structure of the paper is as follows. In Section 2 we give a brief comparison of our approach with other ongoing research work. Next, we describe what we expect from a plagiarism detection framework, i.e. what are the types of student tricks it should be resistant to. In Section 4 we give an overview of the proposed framework and introduce the main concepts. Following this, we describe the three components of the framework: the *Front-end module*, the *Simplifier* and the *Comparator*. Section 5 describes the prototype implementation of the framework for

Prolog and SML programs. Next, we evaluate the system and show execution results. Finally, we give a summary of our work.

2 Related work

Several solutions exist for detecting plagiarism in written documents (like iThenticate [16], FindSame [15], CopyCatch [14], SCAM [26] or the new Hungarian portal from the Computer and Automation Institute of the Hungarian Academy of Sciences called KOPI [5]). However, this is not the case for program sources. A reason for this may be that it is widely believed that detecting plagiarism in programs is much easier than in free text. This is because programming languages are formally defined and, as opposed to the case of free text, it is generally assumed that people use only a few tricks to hide the fact of plagiarism.

Alan Parker and James Hamblen in [23] explicitly say that copied software is "a program which has been produced from another program with a small number of routine transformations". These routine transformations include modifying the comments, changing the names of the variables or (in the worst case) changing the control structures (e.g. using `while` instead of `for`). The suggested technique for comparing programs is the following:

1. Get rid of every comment in the source codes.
2. Get rid of every useless new line, white space, etc.
3. For each pair of source programs use a normal UNIX diff program, which compares the files line by line.
4. Examine the results.

In [8] J. A. Faidhi and S. K. Robinson suggested a scale which defines the level of plagiarism (L0-L6) based on what kind of modifications the cheater used. For example, we obtain L1 from L0 by modifying the comments, L2 from L1 by further modifying the variable names as well, etc. This scale is often used by programs for plagiarism detection to "position" themselves.

Most existing software solutions are based on statistical or lexicographic approach where, for example, they compare identifiers with identifiers to determine how similar the source programs are. Such systems are the DUP [2], SIM [9], SIFF [3] or Bandit [28].

On the other hand, approaches based on structural properties were already proposed several decades ago. For example, in [4] J. M. Bieman and N. C. Debnath suggested building program graphs, while T. J. McCabe proposed [20] to compute a characteristic numeric value, a metric, for each program code according to its complexity (which was based on the number of computation paths available within the program). This metric is widely known today as *cyclomatic complexity*.

Further programs that support structure comparison include the *Plague* [29], the *YAP* (Yet Another Plague) series [30], and the *Moss* (Measure Of Software Similarity) [25] program. Plague builds so called structure profiles for source codes

and compares them. The YAP programs implement a two phase approach. First they convert the source programs into a more unified form, e.g. removing comments, translating upper-case letters to lower case. In the second phase (depending on the actual YAP version) they apply algorithms, such as Heckel's isolation technique [12], that are resistant to specific structural changes, e.g. changing the order of independent statements or replacing a procedure call by the procedure body. The authors of Moss have developed a general algorithm for calculating a so called fingerprint from an arbitrary document which they claim to be especially precise in case of source programs.

Paper [21] introduces an XML-based model called XPDec (XML Plagiarism Detection Model) suitable for programs written in a procedural language such as C or Pascal. XPDec uses XML to represent structural properties of the source programs and is useful for detecting common forms of reordering plagiarism. An extended version of this approach is presented in [22] which takes also into account the structure of the control sequences in the source programs.

Plagiarism detection is also closely related to code duplication detection. Here, the idea is to detect when developers use previously existing code which solved a problem similar to the one they are currently trying to solve. This may indicate a design problem as the duplicated code is difficult to maintain (e.g. fixing bugs must be done in several places). Although most of the existing solutions for duplication detection are based on the lexicographic approach, some of them use the structural properties of the source codes [24, 18].

Our approach introduced in this paper belongs to the group of program plagiarism systems utilising the structural properties of the source programs. However, instead of providing sophisticated comparison techniques that are resistant to the most common tricks we apply so called reduction steps to create more abstract views of the programs. These views are then compared by using relatively simple comparison algorithms. We argue that this approach makes our system fairly efficient and easy to customise.

3 Goals

We now discuss the most significant student tricks we believe a plagiarism detection framework should be resistant to. To illustrate such tricks in a language independent way we use pseudo-language examples below. We realise that there exist tricks only applicable for specific programming languages. Handling these is the task of the concrete implementation of such a framework (cf. Section 4).

Changing the names of identifiers and variables is the most common trick. A piece of source code which contains only single letter variable names may look rather confusing and tangled. However, it can be easily transformed into a program which uses talkative names. For humans, sometimes only this is enough to hide the fact of plagiarism. A similar trick is to change the natural language in which the program identifiers are formulated: use English names in one program and use another language in the other. It is also possible to change not just the variable, but the

function and/or predicate names, too. For example, it is very easy to transform the function

```
void solve_the_problem(Input_data, Results) {...}
```

... to the following:

```
void do(Input, Output) {...}
```

One can also change the number of arguments (the so called *arity*) of the functions, without affecting the code. For example one can use dummy parameters, which are set to something irrelevant at call time. If one changes not only the name of a function, but also its arity, it may become really difficult for the human to recognise that it is semantically equivalent to some other function.

Sometimes it is profitable for students to cut the code into several pieces and place them into separate files using the module system of the given language. Similarly, reordering the sequence of the function definitions in a source file is an easy, but often effective trick. Students also like to change the order of statements in the body of a function if these statements do not depend on each other: for example, two independent variable assignments can be switched. In case of logic programming languages, this kind of trick is very common as a body of a predicate is the logical conjunction of so called goals. This means that these goals can often be reordered freely without effecting the execution of the program.

Putting useless functions into the code may also be used to disguise plagiarism. For example we can "borrow" some code from another program which has nothing to do with the current programming assignment. Computer based methods may find this disturbing, because this technique introduces new variables and functions, changes the size of the file, etc. Sometimes one can recognise this trick by doing static source code analysis and detecting that these functions are never called, but this is not true in general.

Consider the following example, where the procedure `calculate` will never be called. This procedure can be anything, most likely a piece of some big code, with the only aim to conceal the fact that the original source code for `solve_the_problem` was made by some other individual.

```
int solve_the_problem(A, B) {
    if (A > 0) {
    ...
    X  = A + 35;
    ...
    if (X < 0)                    // X cannot be negative here
      calculate(X, B);
    else
      X = 2;
    ...
    }
    ...
}
```

In the general case those parts of the program which are never called can only be detected at run time. Unfortunately, even if we detect such code fragments it does not mean that we found an instance of plagiarism. Sometimes such code is simply the result of programming errors, which even the author of the program is not aware of.

Analogously to placing useless procedures in the program code one can place useless calls in the body of a procedure without changing its task. In the following example we show two totally useless lines inserted into a function, not changing the execution of the program:

```
...
C = 2; A = 3-C;                    // A = 1
...
if (C == A+1) {                    // check if C = 2
...
```

Finally, we show two tricky, but easily implementable types of program transformation. The first we named *call-tunneling*, while the second *call-grouping*. Call tunneling is based on the idea that instead of letting function `A` to call function `C` directly, we insert an intermediate function `B`. In this new scenario `A` calls `B` and `B` calls `C`. If function `B` returns what it got from `C` without any modification, then the transformed program will be equivalent to the original one. Call-tunneling is very hard to detect, because, for example, function `B` is actually called during the execution, therefore it seems to be an important part of the program.

Call-grouping is a simple technique to significantly modify the structure of a program even if one does not really understand what the code actually does. The main idea is very similar to that of call-tunneling: if there is a function which calls several others, we can regroup these calls into some new functions to produce a totally different code structure. Let us consider the following piece of code:

```
int original_function(A, B) {
  T = call1(A);
  Q = call2(B, T);
  E = call3(Q);
  Z = call4(A, E);
  return call5(Z);
}
```

Using call-grouping one can transform it to the following equivalent program.

```
int groupped_function(A, B) {
  E = temp1(A, B);
  return temp2(A, E);
}
```

```
int temp1(A, B) {
  T = call1(A);
  Q = call2(B, T);
  return call3(Q);
}

int temp2(A, E) {
  Z = call4(A, E);
  return call5(Z);
}
```

Notice that functions `call1`,...,`call5` are invoked in the same way as in `original_function`, but two new grouping functions are also introduced.

4 The framework

The proposed framework consists of three main components which are handled by independent program modules:

1. *Front-end:* performs source code to model *mapping*
2. *Simplifier:* carries out model *reduction*
3. *Comparator:* does model *comparison*

The *Front-end* creates a mathematical entity — which we call a **model** or an **abstract view** — from the source program to be examined. Subsequently, these views can be reduced in many ways by the *Simplifier*, creating different **abstractions** of the original model. Having the abstractions of two source programs, we use the *Comparator* to compare models on the same abstraction level and determine a similarity degree (a number between 0 and 1). As the abstraction becomes higher, the similarity of the abstract views is less and less indicative of the similarity of the original programs. Therefore we assign a factor (again a number between 0 and 1) to each abstraction level, with which we multiply the similarity degree obtained earlier.

Figure 1 shows the overview of the proposed framework. Here we start from two source programs `source A` and `source B`. The *Front-end maps* these sources to two models, `model A` and `model B`. Higher and higher abstractions of these models are produced by *reductions*, using the *Simplifier*. Finally, the models on the same abstraction levels are compared with each other.

In the following subsections we discuss in detail the main parts of the framework.

4.1 Source code to model mapping

In general, the entity to which a program source is mapped can be chosen arbitrarily. For example, let us consider the size of the program source (e.g. in terms of characters used) as an abstract entity characterising the program, and consider the advantages and disadvantages of this choice. It is true that if we

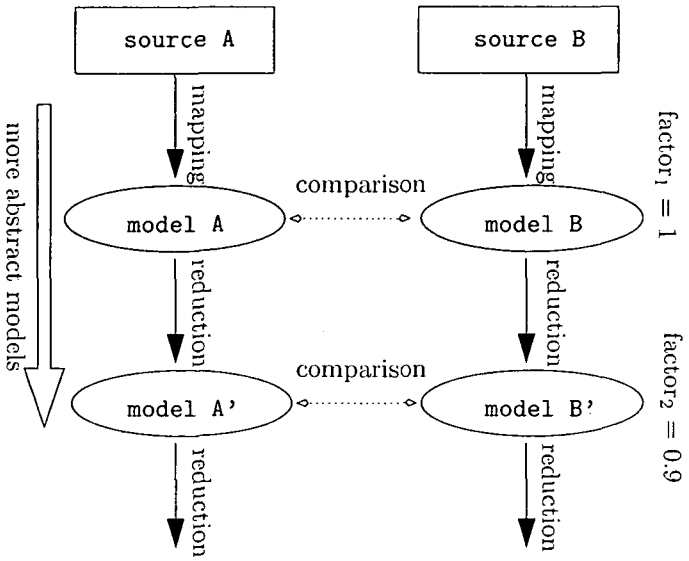


Figure 1: Overview of the proposed framework

examine two entirely identical programs, then the comparison of their abstract views will signal match (the sizes of the programs will be the same). It also sounds feasible to consider the two program instances suspicious, if their size, in terms of characters, is exactly the same. However, if the programs are similar, but not identical, then the program size abstraction cannot give any hint on their similarity.

A further issue is that of simplifying transformations. When a program is characterised by its size, practically no further simplifications can be applied. The only, very weak option is to make further abstractions by rounding the size, e.g. using 1 kbyte instead of 1324 bytes.

Therefore, the abstract view must be more sophisticated (to allow diverse abstraction levels) and, more importantly, it must be possible to draw conclusions on the similarity of the programs from the similarity of the abstract views.

Therefore we suggest the use of *directed, labelled graphs* as the abstract views characterising the programs. Here the meaning of nodes, edges and their labels may vary from implementation to implementation. For example, the abstraction may be the program call graph, the data-flow graph of an execution, or — in case of object-oriented languages — the graph describing the object structure. The labels are used to describe the properties of the nodes and edges, e.g. to express that a node represents a built-in entity and not a user function.

Note that we suggest to ignore the labels in graph comparison, as we would like the similarity measure to focus on the graph structure. A further benefit of this is that it makes the comparison algorithms simpler and faster. However, the reduction steps do use the information stored in labels. This may result in somewhat strange

effects: two graphs, that are considered isomorphic on one abstraction level, become non-isomorphic on the next level, provided the given reduction step uses the labels.

The graph representation is general enough to describe any kind of entity. As an extreme, even our first example, the program size abstraction, can be described as a labelled graph (with a single node whose label is the size).

4.2 Model reduction techniques — abstraction levels

One can envisage some kind of perfect mathematical models, that contain every bit of information present in the program source code. In this case we can be sure that, when two such perfect models are isomorphic, the corresponding program source code is the same. Of course, such a model is nothing else, but the source code itself in a different representation.

For any programming language and for any specific piece of source code, the lowest abstraction level, which we call level 0, could be considered to contain perfect models only. At first, one may think that the best one could do is to directly compare such perfect models. However, this may require a very sophisticated comparison algorithm, which is on one hand fast and easy to customise, and on the other hand resistant to the possible cheating methods mentioned in Section 3. Instead, we decided to follow a different approach using a series of views with increasing abstraction levels.

We thus propose to use several abstraction levels (as shown in Figure 1) and use *relatively simple and fast comparison algorithms* between models on the same level. Higher abstraction levels are built from lower ones (possibly utilising the labels in addition to graph structure) using certain transformations, called *reduction* steps. Our task is to transform the initial perfect models to ones which are more and more resistant to specific tricks, and which still represent the original program sources as much as possible.

Naturally, reduction steps are destructive operations: with every bit of dropped information we widen the gap between the perfect model and the model in question.[3] Because of this, a perfect match (isomorphism for example) between two models on a high abstraction level "means less" than the same type of match on a lower level. To handle this, we assign a factor to each abstraction level in question, with which we multiply the similarity degree achieved on that level.

We define the similarity of two programs as

$$\max_{1 \le i \le n} F_i S_i \tag{1}$$

where n is the number of abstraction levels in the concrete implementation of the framework. F_i is the factor assigned to abstraction level i (a number between 0 and 1) and S_i is the actual similarity degree obtained on abstraction level i (also a number between 0 and 1). We require that $F_{i+1} < F_i$ holds for any i, i.e. the factors

[3] In theory we may end up in a point where every model becomes a singleton graph (a graph consisting of a single node): on this level every pair of models is isomorphic.

1. $i = 1, Max = 0$
2. compare the two models on abstraction level i, i.e. calculate S_i
3. calculate $Max = max(S_i * F_i, Max)$
4. in case of isomorphism ($S_i = 1$), exit with the output value Max
5. if $Max \geq F_{i+1}$, exit with value Max, otherwise $i = i + 1$ and goto step 2

Figure 2: The algorithm for determining the similarity degree of two programs

assigned to the abstraction levels form a strictly monotonic decreasing sequence. However, we do not pose any restrictions on the values S_i and S_{i+1}.

To determine the maximum, we may simply calculate expression (1). For example, let us assume we have two abstraction levels, level 1 and 2 with factors 1 and 0.9 respectively. If our models are assigned a 98% similarity on level 1 and are isomorphic on level 2, the algorithm calculates the values $0.98 * 1 = 0.98$ and $1 * 0.9 = 0.9$ respectively. The final result is the larger of these, namely 0.98.

We can optimise this naive algorithm in the following way. Whenever we detect that the maximum value we may obtain in the next abstraction level (which is F_{i+1} as $S_{i+1} \leq 1$ holds) is less or equal than the current maximum value Max, we can stop. This is because the factors are decreasing, thus for every $j = i + 1, \ldots, n$ it holds that $F_j * S_j \leq Max$. This trivially means that if we detect isomorphism between two models at abstraction level i we can immediately finish execution. When we stop, the final result (i.e. the similarity degree of the source programs in question) is the current maximum Max. This algorithm is shown in Figure 2.

4.3 Model comparison algorithms

In Section 4.1 we argued that directed, labelled graphs are good mathematical constructs for describing models of programs. Considering this, the concrete comparison algorithms are most likely related to graph theoretical algorithms.

In general, our task is to define in what extent are two graphs *similar* to each other. Let us first consider the problem of graph isomorphism as an extreme case of graph similarity.

4.3.1 Graph isomorphism

The problem of graph isomorphism is the following. Given two graphs, G and H, we look for bijection f between the nodes of the graphs, so that (x, y) is an edge in G if and only if $(f(x), f(y))$ is an edge in H.

The graph isomorphism problem belongs to the class of NP problems, but we still do not know if it is NP-complete [1]. However, in special cases we know the complexity exactly or at least we can produce algorithms which run with acceptable

To calculate the code $C(T)$ of tree T:

1. determine the child subtrees of the root of T, N_1, N_2, ..., N_k
2. determine the codes for N_1, N_2, ..., N_k
3. sort the codes $C(N_1)$, $C(N_2)$, ..., $C(N_k)$ into ascending order using their binary values for ordering. Assuming the concatenation of these produces a sequence S, return the sequence $1S0$ as the code assigned to T

Figure 3: The algorithm for calculating the code of a tree.

speed. For example we know polynomial algorithms for planar graphs as well as for graphs where the maximum vertex degree is bounded |7|.

In case of trees a more straightforward approach is applicable |17|. Namely, it is possible to construct a code in *linear time* for two trees T_1 and T_2, which fulfils the following two criteria:

1. if T_1 is isomorphic to T_2, then the code of T_1 equals to the code of T_2
2. if the code of T_1 equals to the code of T_2, then T_1 is isomorphic to T_2

Actually creating a tree code is nothing more than applying a geometrical transformation that maps a 2D tree to a one dimensional sequence of two characters. One can use the digits "0" and "1" or the parentheses "(" and ")" as the elements of the sequence, and accordingly the code of a leaf is "10" or, when parentheses are used, "()". Let T be the tree to be encoded and let $C(T)$ denote the code assigned to the tree T. Now, the recursive algorithm presented in Figure 3 assigns a binary number to any tree T.

Two examples of such encoding are given in Figure 4. We note that sometimes it is useful to apply a special notation for the leaves, to distinguish these from the code corresponding to other parts of the tree. We will use letter L for this purpose. Accordingly, the codes in Figure 4 can be written as 1LL0 and 1L1LL00, respectively.

It is important that the algorithm in Figure 4 can also be used for DAGs (Directed Acyclic Graphs), i.e. directed graphs, not containing directed circles. In this case, in addition to a DAG, the algorithm requires that a "root node" is specified, which serves as a starting point for the algorithm. An example of such a graph (the starting node is denoted by R) and its code can be seen in Figure 5.

Note that the code of a DAG can also be obtained by first transforming the DAG into a tree, and then taking the code of this tree. The transformation takes a vertex a with $n > 1$ incoming edges and replaces it by n new vertices, each with a single incoming edge (and each new vertex inherits all the outgoing edges of the original one). By repeating this transformation step we can eliminate all vertices with multiple incoming edges and thus obtain a tree from the DAG. The right hand

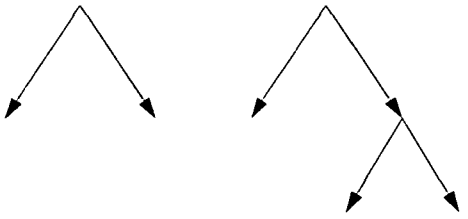


Figure 4: Two examples for the coding scheme

side of Figure 5 shows the result of this transformation process, when applied to the DAG on the left hand side.

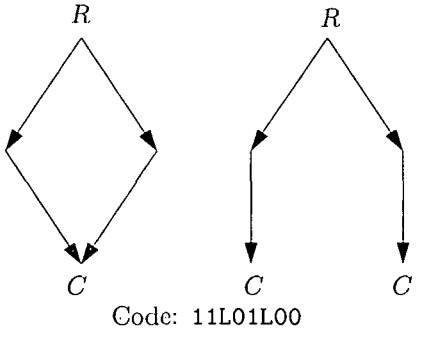


Figure 5: A DAG and the corresponding tree with its code

4.3.2 Graph similarity

Checking isomorphism is usually not enough by itself. The reason is that we cannot expect (at least on the lower abstraction levels) that the program graphs will be totally isomorphic, even with the most sophisticated source code to model mappings and reduction techniques. Actually we would like to detect if two graphs of hundreds of nodes (which are very typical for the programs we use) at a given abstraction level are *nearly* identical.

The general approach we use is to check how it is possible to transform one DAG code to another. For example, let us consider the following two sequences that correspond to the codes in Figure 4.

First sequence (F): `1L1LL0`

Second sequence (S): `1LL0`

The transformation steps we need to e.g. convert the first sequence into the second one can be described as follows: "remove 1 and then L from position 3 and 4 in sequence F". Such transformation steps can be constructed by e.g. using algorithms solving the so called *longest common subsequence (LCS)* problem [13]. Let us denote by $\Delta(A, B)$ the transformation steps between two arbitrary sequences A and B. $\Delta(A, B)$ is a set, containing pairs. The first part of such a pair can be 0, 1 or L, the element to be added or removed. Each of these elements is preceded by either a plus(+) or a minus(-) sign, corresponding to element addition and removal. The second part of a pair is an integer, describing the position the specific transformation step should be applied at. In our case, $\Delta(F, S)$ is the set $\{(-1, 3), (-L, 4)\}$.

To determine the similarity degree of two arbitrary codes we assign penalties to the specific transformation steps. For example, we may say that the removal of a leaf (i.e. a -L in the transformation set) reduces the similarity by a certain amount, let us say by 0.01. Using the penalties we calculate $\Omega(A, B)$, the discrepancy function describing to what extent codes A and B are different:

$$\Omega(A, B) = min\left(1, \sum_{(E,\ldots)\in\Delta(A,B)} P(E)\right) \tag{2}$$

Here P is the penalty function that assigns a value to a given type of transformation step. Using $\Omega(A, B)$ we define the *similarity degree* of graphs A and B as:

$$1 - \Omega(A, B) \tag{3}$$

Let us note that what we actually calculate here is a variant of the so called Levenshtein and edit distances. The Levenshtein distance [19] between two strings is the minimal number of operations needed to transform one string into the other. By operation we mean an insertion, deletion, or substitution of a single character. The edit distance [6] is a generalisation of the Levenshtein distance in that the operations have costs assigned to them, similar to the costs we have defined above.

4.3.3 Distinct paths in the graphs

Unfortunately, in a special case very similar graphs are considered to be far from each other according to the similarity degree introduced above. The reason for this is the way how DAG codes are built. We have seen in Figure 5 that the code corresponding to node C appears in the code of the whole DAG twice (i.e. we have two L characters in the code, although the original DAG has only one leaf). In general, for any DAG G, the code of a node v appears in the code of G exactly m times, where m equals to the number of distinct paths from the root to v.

Let us assume that the DAGs corresponding to programs A and B differ only slightly in a single node, which is, however, accessible from the root along many distinct paths. Because of the reasons outlined above, the DAG codes corresponding to A and B will differ significantly.

We suggest two ways to overcome this shortcoming, both of which are used in our prototype system described in the next section. First, we suggest to use

reduction steps which decrease the number of distinct paths from the root, thus making the graphs more "tree like". For example, filtering multiple edges in a graph (cf. Section 5.2) reduces the number of distinct paths significantly.

As a second solution we suggest to introduce a slight modification of the similarity degree as defined in (3). This modification relies on identifying nodes with numerous incoming edges (let us call these nodes *popular*)[4]. Using structural decomposition we suggest to calculate a variant of the discrepancy function Ω specified in Equation 2, called Ω'. The final similarity degree will thus be $1 - \Omega'(A, B)$.

We now describe how to compute the value of $\Omega'(A, B)$ for arbitrary two graphs A and B. We assume that a popular node N in A can be associated with its counterpart M in B. In our implementation we use a very simple heuristics for this: we pair those popular nodes whose number of incoming edges and number of arguments are the closest (proving the mathematical properties of this heuristic is a future work). If such a pairing is not possible (if at most one of the graphs contains popular nodes) then $\Omega'(A, B)$ simply equals to $\Omega(A, B)$. If a pairing is possible then we first apply our algorithm recursively to the subgraphs rooted at N and M, i.e. we calculate the value $L = \Omega'(N, M)$. Next, we create DAGs A' and B' from the original ones by replacing N and M by single nodes, having no outgoing edges. The modified discrepancy is then calculated recursively as

$$\Omega'(A, B) = \Omega'(A', B') + L \tag{4}$$

The algorithm introduced above is summarised in Figure 6. In our implementation we offer the user a choice of the discrepancy function (Ω or Ω') through the graphical user interface (see Section 5.5).

```
Ω'(A,B) =
  if pairing_popular_nodes_possible(A,B) then
     (N,M) = pair_popular_nodes(A,B);
     A' = reduce(A, N);
     B' = reduce(B, M);
     return Ω'(A',B') + Ω'(N,M);
  else
     return Ω(A,B);
```

Figure 6: The DAG discrepancy algorithm with popular nodes.

5 The prototype implementation

In the following we present our test implementation of the framework, the *Match* plagiarism detection tool. The current version of Match supports two Front-ends,

[4]In our implementation (see Section 5) nodes with at least 10 incoming edges are considered popular.

for Prolog and SML, as we teach these two languages as part of a Declarative Programming course, and major assignments must be written in these languages. In this paper we describe the Prolog Front-end only, more about the SML interface can be read in [10].

In the following we discuss the implementation details of the relevant parts of the comparison framework (Front-end, Simplifier and Comparator), then we describe the graphical user interface of the system.

5.1 Mapping source code to a model

We chose *call graphs* as the models of Prolog programs. A call graph is a graph where the nodes correspond to the Prolog predicates and the edges to the calls. If in the body of predicate `A` there is a call to predicate `B` then an edge between the nodes `A` and `B` is present in the graph. When there are multiple calls, multiple edges are present. We decided to exclude the built-in predicates (such as `is/2`) from the graph, because they do very elementary tasks and would increase the graph size considerably, without increase in the precision of the model. The graph includes, however, the library predicates and also reflects implicit meta calls, made by using `findall/3`, for example.

Furthermore we made some simplifications to our model: we remove from the call graph the *self-loops*, which correspond to recursive calls. This is because explicit recursion is so common is Prolog that for us it does not contain valuable information. We also remove *back edges* (i.e. edges which point to an already visited node during a depth-first search) in order to avoid cyclic graphs, so that we can work with DAGs instead of general graphs. Although this means that simple reordering tricks can change the resulting graph (as they change the order in which a depth-first search visits the nodes) we do not consider this a problem. This is because in our model, circles actually correspond to so called mutual recursion (for example when `A` calls `B` and vice versa). Our experience, however, is that mutual recursion is very rarely used by students and so neglecting it does not effect the final similarity measure in a significant way.

Finally, those predicates to which there was no reference in the source code are not included in the graph, i.e. the call graph consists of a single component.

Call graphs are well suited for Prolog programs. This is because the only control structure of Prolog is the predicate invocation, it lacks `while`, `for`, `goto` or any other "usual" imperative control elements.

The call graph is built from the program source code by using static source code analysis. For this we slightly modified the `xref` package of SICStus Prolog.

5.2 Model reduction techniques

For the comparison of Prolog programs we have defined four reduction steps, which are applied in succession. This means that the comparison can be performed on five abstraction levels. To each reduction step we have assigned a similarity

factor as introduced in Figure 1. These factors were chosen empirically. The actual reduction steps and their similarity factors are the following:

1. *non-called predicates:* we remove those predicates which were not called during a test call. The test call is provided by the tester. Test calls are usually simple tests which are easily solvable by the programs. The assigned similarity factor is 0.95.
2. *library or dynamic predicates:* here we omit the library and dynamic predicates from the call graphs. The similarity factor is 0.9.
3. *multiple edges*: we remove multiple edges from the call graph and only keep one edge between any given two predicates. This helps handling the popular node problem presented in Section 4.3.3. We set the similarity factor to 0.8.
4. *topological isomorphism:* we remove those vertices from the call graphs which have a degree of 2 (one incoming and one outgoing edge). This helps to detect the call-tunneling trick. The similarity factor here is 0.7.

Actually the user of the plagiarism detection system can decide to skip some of the reductions steps (cf. Section 5.5), which results in less than 5 abstraction levels. In this case the factors can be different from the ones shown above, as the value of a factor is usually set in a "context sensitive" way, considering what other reduction steps have been done previously. For example, in our concrete implementation, if we use the first, second and fourth reduction step, but we do not filter multiple edges, the factor for the highest abstraction level (the one corresponding to step four above) is set to 0.85. Thus step 4 when applied after steps 1 and 2 is considered to be slightly less "destructive" than step 3 in the same context (the corresponding factors being 0.85 and 0.8, respectively). Our experience is that using step 3 and 4 simultaneously has significant cumulative effect, justifying the similarity factor of 0.7, when all the above reduction steps are applied.

5.3 Model comparison algorithms

In our system the comparison of models is based on the coding technique and the similarity degree introduced in Section 4.3. We have chosen this approach because we found that comparing codes often gives a good intuitive characterisation of the similarity of the programs.

For example, if the codes match the corresponding models are trivially the same. If one of the codes contains the other as its subsequence, then it can be suspected that one student got the other's program and added some new structure to it.

Call grouping can also be detected from the codes[5]. For example if predicate `P` calls `T` which calls four other predicates, the corresponding code will be `( L (` `L L L L ) )`, where parentheses represent binary values and `L` represents a leaf as described in Section 4.3. If we apply call grouping, for example `T` will call `Q` and `W`, each of which will call two other predicates, then the code takes the form `( L ( (`

[5] We will use parenthesis in the codes below, instead of binary digits, as this makes call nesting more apparent.

`L L ) ( L L ) ) )`. Here the second, third, fourth and fifth parenthesis has to be removed in order to get the original code.

We use the widely available UNIX `diff` program to actually enumerate the differences between the codes A and B, i.e. to calculate $\Delta(A, B)$. The `diff` program uses a variation of the LCS algorithm (cf. Section 4.3.2). The way we make use of `diff` is the following. First we make two files corresponding to the two codes we would like to compare. The way we create the content of such a file is the following: each `(`, `)` and `L` in the tree code is put on a separate line. For example the file corresponding to the code `(LL)` shown in Figure 4 will contain four lines:

```
(
L
L
)
```

Next, we let the UNIX `diff` utility calculate how these files can be made equal, i.e. to produce the instructions on which leaves and nodes should be added or removed to make call graphs A and B isomorphic. Actually we always try to modify the bigger graph (i.e. the graph with longer code) and check what transformations we can use to obtain the smaller one.

By analysing the information given by the `diff` utility we assign a similarity degree to the pair of codes. As we described in Section 4.3, we start with degree 1 and for each difference we subtract a "penalty" fraction, which reflects how much we should "punish" the given modifications of the code sequences. This corresponds to calculating equation (3) in Section 4.3.2. We found that the penalties shown in Table 1 are very usable[6]:

Type of modification	Penalty
removal of a leaf	0.01
addition of a leaf	0.03
removal of a node	0.02
addition of a node	0.06

Table 1: Penalties used by the Match system.

Accordingly, if we need to remove one leaf from our bigger call graph to make it identical to the smaller one, then the similarity degree is 0.99. As one can see, addition is always penalised more than removal. This is because of our experience that cheating students usually try to copy and modify the work of a fellow student

[6]Note that the addition and removal of a node actually corresponds to two differences, one for the opening and the other for the closing parenthesis.

while keeping the original parts intact. This way the original program will be part of the resulting (bigger) program. So, our assumption is that in case of plagiarism the bigger graph can be reduced to the smaller one by applying node and edge removals.

According to this assumption, we actually offer to use the `diff` program in two different modes. The mode named `sdiff` (simple `diff`) means that we only consider those transformations that require only node or edge removals from the bigger graph. Otherwise we conclude that no plagiarism happened. In the other mode, called `fdiff` (full `diff`), we make no such assumption. This results in more false positive results, but it also increases recall significantly (see Section 6).

Having explained the concrete implementation, we reiterate the issue of abstraction levels. Let us consider two graphs which differ only in the multiplicity of the edges. In the absence of abstraction levels, using `diff` alone, we could easily get a similarity degree of 0, provided there is a sufficient number of multiple edges in the graph. However, when the multiple edge removal abstraction is applied we get a similarity of 0.9, which may be more appropriate. This shows that the introduction of abstraction levels is a useful extension, in addition to the `diff` algorithm.

We have also made a further improvement in the calculation of the similarity degrees, as discussed in the following subsection.

5.4 Generating mappings between predicates

Although by calculating the similarity degree of the source programs and presenting the user the most promising pairs we have already fulfilled our original goal, it greatly helps the user of the system if we present some kind of a "proof" of cheating as well. We produce such a "proof" in the form of a mapping between the predicates of the two programs. In this mapping a predicate of one program is paired with the predicate of the other program which is most similar to it. This is a very useful guide to the user when she verifies the results manually.

These mappings are generated by a systematic deterministic traversal of the codes in question. We start from the nodes corresponding to the root predicate (i.e. a predicate which is the entry point of all the student programs). These nodes are paired with each other. Then we visit the neighbours of the starting nodes and pair them using their codes. We continue this algorithms recursively. Whenever there is ambiguity, e.g. we are examining two nodes with multiple neighbours having the same codes, we use a heuristic: those nodes will be paired whose number of arguments differ the least. Note that the mappings are actually derived from the models belonging to the abstraction level where the maximal similarity degree was found. An example mapping is shown in Figure 8.

As mentioned above, the "quality" of the mapping is also taken into consideration when calculating the similarity degree. In the current implementation we actually decrease the similarity degree of the programs by 0.005 for each pair of predicates mapped to each other, which have different arities.

5.5 The graphical user interface

Match offers a graphical user interface (GUI) where the user can customise the parameters of the comparison and can view the results.

A screenshot of the main window can be seen in Figure 7 (it shows the state of the system right after a successful execution)[7]. On the top of the window we find four buttons. The first one called "Make info" invokes the Front-end, i.e. it creates the models from the source codes. This practically means that Match searches for source programs in the given directory and for each source code it creates a special file containing the labelled call graph.

These files are loaded by the second button named "Load info". In the bottom of the window we can see that in this specific example we loaded 32 such graphs.

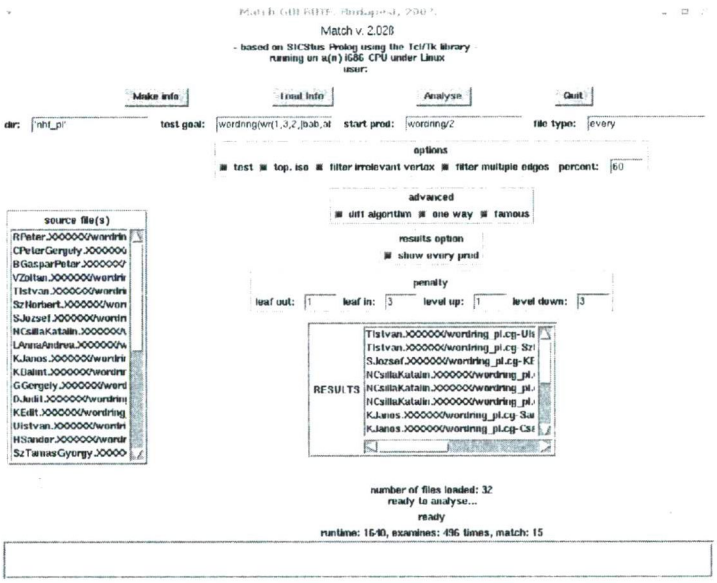


Figure 7: The Graphical User Interface of the Match system

The third button called "Analyse" starts the comparison process based on the parameters the user specified by using the check boxes located in the "options" and "advanced" areas. These options basically tell Match what kind of reduction steps it should apply, whether it should use the `diff` algorithm[8] and what is the similarity threshold (i.e. only hits with similarity degree greater than the threshold will be presented). In the example shown in Figure 7 we selected all the reduction steps,

[7]Note that we changed the name of the students because of privacy issues.

[8]If this option is disabled, then isomorphism is used instead of similarity, i.e. the similarity degree of the models is considered to be a binary value: 1 means that the models are isomorphic, 0 means they are not.

asked Match to use `diff`, take special care of the popular nodes (called famous in the GUI) and set the threshold to 60%.

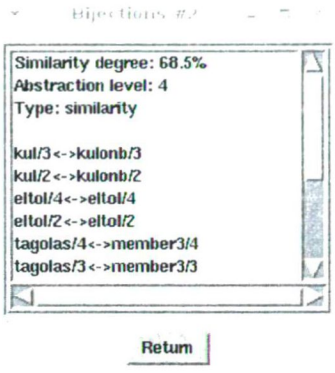


Figure 8: A mapping between predicates proving the fact of plagiarism

When we use the `diff` algorithm, we can also set the penalties (in units of 0.01) the Match program should use to determine the similarity of the two graphs. In this example the penalties are set to the ones described in Section 5.3.

After a successful execution the suspicious pairs of programs are shown in the middle of the screen under the title "Results". In Figure 7 we have 15 such pairs. If we select one of these pairs, Match displays the predicate mapping between the two source programs. Namely, we can see which predicate in one program matches which predicate in the other, as shown in Figure 8. In this concrete mapping we can see that, for example, predicate `kul` corresponds to predicate `kulonb`, both of them having 3 arguments. We can also see that the similarity of these programs is calculated to be 68.5%, and that this was found on abstraction level 4. The next line, "Type:", indicates that even on this abstraction level the codes were only "similar", i.e. non-isomorphic.

6 Evaluation

Below we first reiterate our goals presented in Section 3 and examine how they are fulfilled by the Match system. Next, we present real life execution results showing that the framework convincingly detects plagiarism in student programs.

Table 2 summarises the student tricks we have described in Section 3 and for each gives a brief explanation of how the given trick was handled in the implementation of the plagiarism detection framework.

We now proceed to discuss the performance evaluation of the Match system. We were lucky enough to have abundant amount of Prolog source programs to test the prototype on. Moreover, several students were kind enough to provide us with some hints on what cases were they cheating (*after* they completed the course and

Student trick	Preemptive measure
changing the names of the identifiers	we do not store names in our models
changing the arity of the functions	arity is not used in model comparison
splitting the program into several modules	module boundaries are not taken into account
reordering the function definitions	the call graph is not affected
reordering the statements within a function	the call graph is not affected
putting useless functions in the program	using the "non-called predicates" reduction step
putting useless calls in a function	using the "non-called predicates" reduction step
call-tunneling	using the "topological isomorphism" reduction step
call-grouping	using the `diff` algorithm

Table 2: Our goals and the way how they are achieved.

were promised full amnesty). So we had the minimal expectation that the Match system should at least mark those assignments as matched pairs.

The 73 source programs[9] were evaluated against 4 different similarity thresholds. For example, the 60% threshold means that our system shows pairs of source codes which have similarity degree at least 60% percent. For every threshold, the system was run with 18 different parameter variations. These include the most useful settings in practical cases. These 18 variations are based on the following 6 basic variants:

variant B:	all options are disabled (base case)
variant N:	filtering non called predicates
variant NL:	N + filtering library/dynamics predicates
variant NLM:	NL + filtering multiple edges
variant NLT:	NL + topological isomorphisms
variant NLMT:	NLM + topological isomorphisms

In the first six cases we do not use `diff`, i.e. we only consider graph isomorphism

[9] We had actually 92 submitted Prolog homework, but from these we excluded those programs that either do not compile or do not solve the required number of test cases.

between the models at different abstraction levels. The next twelve variations are obtained by applying `diff` in two different modes, simple and full (cf. Section 5.3).

	Execution time	Compared pairs	Hits	Relevant hits	Recall	Precision
B	15.40s	2628	6	6	46%	100%
N	20.33s	2628	6	6	46%	100%
NL	30.11s	2628	6	6	46%	100%
NLM	31.50s	2628	6	6	46%	100%
NLT	31.28s	2628	6	6	46%	100%
NLMT	33.40s	2628	16	8	61%	50%
B + sdiff	60.13s	2628	8	6	46%	75%
N + sdiff	78.07s	2628	8	6	46%	75%
NL + sdiff	124.41s	2628	17	8	61%	47%
NLM + sdiff	179.78s	2628	22	8	61%	36%
NLT + sdiff	162.34s	2628	17	8	61%	47%
NLMT + sdiff	364.04s	2628	32	10	76%	31%
B + fdiff	98.2s	2628	48	10	76%	21%
N + fdiff	122.02s	2628	48	10	76%	21%
NL + fdiff	180.78s	2628	65	11	84%	17%
NLM + fdiff	295.6s	2628	70	11	84%	16%
NLT + fdiff	232.60s	2628	65	11	84%	17%
NLMT + fdiff	414.43s	2628	80	13	100%	16%

Table 3: Match results for the threshold of 60%.

For every setting we measured the run time, the number of hits and the ratio of hits and the relevant hits (precision). To determine the relevant hits we examined the program pairs manually, and decided if the given case should be considered plagiarism or not. We also made a serious manual effort to check if there were any cases of plagiarism that were not found by Match. We concluded that the 13 pairs discovered by the most complex run of Match were the only cases where one program code was derived from the other one.

Assuming that the number of all hits is 13, we calculated the so called *recall* which is the ratio of the number of relevant hits returned by the program and the number of all hits.[10]

The results for threshold 60% are listed in Table 3. The test were run under Linux on Intel Celeron 450Mhz processor with 128 MByte of RAM.

From our tests we can draw several conclusions. First, we found that students do not often use sophisticated tricks. This can be seen from the fact that adding more reduction steps does not significantly improve the effectiveness of Match: most of the cheaters are caught already at the lower levels. At the same time further reduction steps do result in new hits, so higher abstraction levels are by no means useless. For example, 80% percent of the cheaters found in the second block were already uncovered after two reduction steps (variant NL). These included the programs of a pair of students who claimed they worked on the modifications for more than 5 hours, and in spite of this, their similarity degree was nearly 90%.

We can also see that although precision drops back significantly as we use more and more abstraction levels, the results are still acceptable. In the worst case (here the precision is 16%) one in six pairs of codes is a proper hit among 70 suspicious pairs. Although this requires some effort from the person verifying the results of Match, the amount of manual work is still almost two magnitudes less than that required when the plagiarism detection framework is not used (over 2500 cases).

We can also conclude that both the abstraction mechanism and the similarity degree calculated in (3) are needed. There were cases when the plagiarism was detected with a high degree of similarity due to the fact that only minor differences were found on abstraction level 1, for example. Without considering graph similarity, we would have needed to use more reduction steps to make these models isomorphic, resulting in a smaller similarity degree. This shows how useful the technique introduced in (3) can be. However, the opposite situation also occurred. Namely, we could find programs (relevant hits) which were isomorphic on a relatively high abstraction level, but calculating (3) on the previous level gave much lower similarity. This shows the significance of using reduction steps.

Our approach is very fast. For 73 programs, checking of all the pairs took between 15.4s (no diff and no abstraction levels) and 414s (when all of the reduction steps were applied and the full diff algorithm was used).

7 Future work

Our future plans include the integration of the most promising statistical and/or lexicographic approaches in the framework. This way we can use hybrid comparison techniques which we hope can be more efficient than the pure structural approach in some cases.

We would also like to develop Front-ends for further programming languages. In case of procedural languages, such as C, Pascal or several script languages, the

[10] Precision and recall are measures widely used for evaluation information retrieval techniques, see e.g. in [11].

control structures are the main bearers of information. When dealing with object-oriented languages, C++ or Java for example, the class hierarchies should also be properly represented in the models.

The heuristics used by the Match system can also be improved. These include the heuristics used when pairing popular nodes and predicates between two models. Furthermore, the user interface lacks some functionality. For example, it is often the case that a larger group of students submit similar assignments. In such a case Match returns all pairs within the group as suspicious. It would help the person verifying the results, if the group of cheaters was identified, as a whole.

Beside homeworks, we would also like to measure the performance of the framework for really large source programs (e.g. millions of lines of codes) as well.

Finally, it would be interesting to extend the Match system with adaptive penalty weights, i.e. to let the system automatically determine the penalty function based on certain properties of the given source programs (size, complexity, etc).

8 Conclusions

In the paper we have presented a plagiarism detection framework which is capable of calculating a similarity degree for a pair of program sources. The framework uses directed, labelled graphs to represent the structural information extracted from the programs. Instead of using sophisticated comparison algorithms our approach combines the use of relatively simple comparison techniques together with simplifying graph transformations, called reduction steps.

We have presented the three main components of the generic framework: the Front-end which converts programs to graphs, the Simplifier, which carries out the reduction steps and the Comparator, which calculates a similarity degree for the graphs. We have described the implementation of the framework, the Match system, which has been successfully used to detect plagiarism in homework assignments for years. We have also presented a detailed performance evaluation of the system.

We believe that the novel architecture of our approach, based on simplifying graph transformations and straightforward comparison algorithms, has proved to be a viable technology for plagiarism detection in source programs.

Acknowledgements

The authors would like to thank the help of Tamás Benkő in the development of the Match system.

References

[1] V. Arvind and Piyush P. Kurur. Graph isomorphism is in SPP. *Inf. Comput.*, 204(5):835–852, 2006.

[2] Brenda S. Baker. A theory of parameterized pattern matching: algorithms and applications. In *In Proceedings of the 25th Annual Symposium on Theory of Computing*, pages 71–80, 1993.

[3] Brenda S. Baker and Udi Manber. Deducing similarities in Java sources from bytecodes. In *Proc. of Usenix Annual Technical Conf.*, pages 179–190, 1998.

[4] J.M. Bieman and N.C. Debnath. An analysis of software structure using generalized program graph. In *In Proceedings of COMPSAC*, pages 254–259, 1985.

[5] Computer and Automation Institute of the Hungarian Academy of Sciences. Online plagium detection portal. `http://www.kopi.sztaki.hu/`, 2007.

[6] M. Crochemore, C. S. Iliopoulos, Y. J. Pinzon, and J. F. Reid. A fast and practical bit-vector algorithm for the longest common subsequence problem. In L. Brankovic and J. Ryan, editors, *Proceedings of the11th Australasian Workshop On Combinatorial Algorithms*, pages 75–86, Hunter Valley, Australia, 2000.

[7] David Eppstein. Subgraph isomorphism in planar graphs and related problems. In *SODA: ACM-SIAM Symposium on Discrete Algorithms (A Conference on Theoretical and Experimental Analysis of Discrete Algorithms)*, 1995.

[8] J. A. W. Faidhi and S. K. Robinson. An empirical approach for detecting program similarity and plagiarism within a university programming environment. *Comput. Educ.*, 11(1):11–19, 1987.

[9] Dick Grune. The software and text similarity tester sim. `http://www.cs.vu.nl/~dick/sim.html`.

[10] Dávid Hanák. Computer based support for teaching declarative languages (in hungarian). Master Thesis, 2001.

[11] David Hawking and Nick Craswell. Very large scale retrieval and web search. In Ellen Voorhees and Donna Harman, editors, *TREC: Experiment and Evaluation in Information Retrieval*. MIT Press, 2005.

[12] Paul Heckel. A technique for isolating differences between files. *Commun. ACM*, 21(4):264–268, 1978.

[13] Daniel S. Hirschberg. Algorithms for the longest common subsequence problem. *J. ACM*, 24(4):664–675, 1977.

[14] Gwyneth Hughes, Sharon Brown, Mike Jakobson, Chris Philpot, Paul Dwight-Moore, Nick Jarrett, Toby Grainger, and Barry Short. Report on the viability of copycatch plagiarism detection software. `http://www.copycatchgold.com/`, 2002.

[15] Digital Integrity. Findsame. `http://www.findsame.com/`, 2007.

[16] iParadigms LLC. iThenticate. `http://www.ithenticate.com/`, 2007.

[17] A. Jovanović and D. Danilović. A new algorithm for solving the tree isomorphism problem. *Computing*, 32(3):187–198, 1984.

[18] Rainer Koschke, Raimar Falke, and Pierre Frenzel. Clone detection using abstract syntax suffix trees. In *WCRE '06: Proceedings of the 13th Working Conference on Reverse Engineering*, pages 253–262, Washington, DC, USA, 2006. IEEE Computer Society.

[19] Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.

[20] Thomas J. McCabe. A complexity measure. *IEEE Trans. Software Eng.*, 2(4):308–320, 1976.

[21] Seo-Young Noh, Sangwoo Kim, and S.K. Gaida. An XML plagiarism detection model for procedural programming languages. In *In Proceedings of the 2nd International Conference on Computer Science and its Applications*, pages 320–326, 2004.

[22] Seo-Young Noh, Sangwoo Kim, and Cheonyoung Jung. A lightweight program similarity detection model using XML and Levenshtein distance. In *The 2006 World Congress in Computer Science Computer Engineering, and Applied Computing*, pages 3–9, 2006.

[23] Alan Parker and James Hamblen. Computer algorithms for plagiarism detection. *IEEE Transactions on Education*, 32(2):94–99, 1989.

[24] Matthias Rieger, Stéphane Ducasse, and Michele Lanza. Insights into system-wide code duplication. In *WCRE '04: Proceedings of the 11th Working Conference on Reverse Engineering*, pages 100–109, Washington, DC, USA, 2004. IEEE Computer Society.

[25] S. Schleimer, D. Wilkerson, and A. Aiken. Winnowing: local algorithms for document fingerprinting, 2003. `http://theory.stanford.edu/~aiken/publications/papers/sigmod03.pdf`.

[26] Narayanan Shivakumar and Hector Garcia-Molina. The SCAM approach to copy detection in digital libraries. *D-Lib Magazine*, 15, 1995.

[27] Jill Suarez and Allison Martin. Internet plagiarism: A teacher's combat guide. *Contemporary Issues in Technology and Teacher Education*, 1(4), 2001. `http://www.citejournal.org/vol1/iss4/currentpractice/article2.htm`.

[28] Adrian West. Coping with plagiarism in computer science teaching laboratories. Computers in Teaching Conference, Dublin, 1995.

[29] G. Whale. Identification of program similarity in large populations. *Comput. J.*, 33(2):140–146, 1990.

[30] Michael J. Wise. YAP3: Improved detection of similarities in computer program and other texts. *SIGCSEB: SIGCSE Bulletin (ACM Special Interest Group on Computer Science Education)*, 28, 1996.

Received 18th July 2007

Acta Cybernetica **19** (2009) 217–230.

Limited Codes Associated with Petri Nets

Genjiro Tanaka*

Abstract

The purpose of this paper is to investigate the relationship between limited codes and Petri nets. The set M of all positive firing sequences which start from the positive initial marking μ of a Petri net and reach μ itself forms a pure monoid M whose base is a bifix code. Especially, the set of all elements in M which pass through only positive markings forms a submonoid N of M. Also N has a remarkable property that N is pure. Our main interest is in the base D of N. The family of pure monoids contains the family of very pure monoids, and the base of a very pure monoid is a circular code. Therefore, we can expect that D may be a limited code. In this paper, we examine "small" Petri nets and discuss under what conditions D is limited.

Keywords: free monoid, Petri net, code, prefix code, circular code, limited code

1 Introduction

Let A be an alphabet, A^* the free monoid over A, and 1 the empty word. Let $A^+ = A^* - \{1\}$. A word $v \in A^*$ is a *right factor* of a word $u \in A^*$ if there is a word $w \in A^*$ such that $u = wv$. The right factor v of u is called *proper* if $v \neq u$. For a word $w \in A^*$ and a letter $x \in A$ we let $|w|_x$ denote the number of x in w. The length $|w|$ of w is the number of letters in w.

A non-empty subset C of A^+ is said to be a *code* if for $x_1, ..., x_p, y_1, ..., y_q \in C$, $p, q \geq 1$,

$$x_1 \cdots x_p = y_1 \cdots y_q \implies p=q,\ x_1=y_1,\ \ldots,\ x_p=y_p.$$

A subset M of A^* is a *submonoid* of A^* if $M^2 \subseteq M$ and $1 \in M$. Every submonoid M of a free monoid has a unique minimal set of generators $C = (M-\{1\})-(M-\{1\})^2$. C is called the *base* of M. A submonoid M is *right unitary* in A^* if for all $u, v \in A^*$,

$$u,\ uv \in M \implies v \in M.$$

*Dept. of Computer Science, Shizuoka Institute of Science and Technology, Fukuroi-shi, 437-8555 Japan. E-mail: tanaka@cs.sist.ac.jp

M is called *left unitary* in A^* if it satisfies the dual condition. A submonoid M is *biunitary* if it is both left and right unitary.

Definition 1.1. Let M be a submonoid of a free monoid A^*, and C its base. If $CA^+ \cap C = \emptyset$, (resp. $A^+C \cap C = \emptyset$), then C is called a *prefix* (resp. *suffix*) code over A. C is called a *bifix* code if it is a prefix and suffix code.
A submonoid M of A^* is right unitary (resp. biunitary) if and only if its minimal set of generator is a prefix code (resp. bifix code) ([1, p.46],[3, p.108]).

Definition 1.2. A Petri net is a 4-tuple, $PN = (P, A, W, \mu_0)$ where $P = \{p_1, p_2, \ldots, p_m\}$ is a finite set of places, $A = \{t_1, t_2, \ldots, t_n\}$ is a finite set of transitions such that $P \cap A = \emptyset$ and $P \cup A \neq \emptyset$, $W : (P \times A) \cup (A \times P) \to \{1, 2, \ldots\}$ is a weight function, $\mu_0 : P \to \{0, 1, 2, \ldots\}$ is the initial marking.

Let $t \in A$, and let $\cdot t = \{p \in P | (p, t) \in P \times A\}$ and $t\cdot = \{p \in P | (t, p) \in A \times P\}$. In this paper we shall assume that a Petri net has no isolated transitions, i.e., no t such that $\cdot t \cup t\cdot = \emptyset$. A transition t is said to be *enabled* in a marking μ_0, if $W(p, t) \leq \mu_0(p)$ for all $p \in \cdot t$. A firing of an enabled transition t removes $W(p, t)$ tokens from each input place $p \in \cdot t$, and adds $W(t, p)$ tokens to each output place $p \in t\cdot$. A firing of an enabled transition t in μ_0 produces a new marking μ_1

$$\mu_1(p) = \mu_0(p) - W(p, t) + W(t, p)$$

for any $p \in P$, denoted by $\mu_1 = \delta(\mu, t)$. A string $w = t_1 t_2 \ldots t_r$, $t_i \in A$, of transitions is said to be a $(firing)$ *sequence* from μ_0 if there exist markings $\mu_i, 1 \leq i \leq r$, such that $\delta(\mu_{i-1}, t_i) = \mu_i$ for all i, $1 \leq i \leq r$. In this case, μ_r is reachable from μ_0 by w and we write $\delta(\mu_0, w) = \mu_r$. The set of all possible markings reachable from μ_0 is denoted by $Re(\mu_0)$, and the set of all possible sequences from μ_0 is denoted by $Seq(\mu_0)$. The function $\delta : Re(\mu_0) \times A \to Re(\mu_0)$ is called a next-state function of a Petri net PN [5.p.23]. We note that the above condition for $r = 0$ is understood to be $\mu_0 \in Re(\mu_0)$. A marking μ is said to be *positive* if $\mu(p) > 0$ for all $p \in P$. A sequence $t_1 t_2 \ldots t_n \in Seq(\mu_0)$, $t_i \in A$, is called a positive sequence from μ_0 if $\delta(\mu_0, t_1 t_2 \ldots t_i)$ is positive for all i, $1 \leq i \leq n$. The set of all positive sequences from μ_0 is denoted by $PSeq(\mu_0)$.
Let $P = \{p_1, p_2, \cdots, p_n\}$. A marking μ can be represented by a vector $\mu = (\mu(p_1), \mu(p_2), \cdots, \mu(p_n))$. For every $t \in A$ the vector Δt is defined by

$$\Delta t = (\Delta t(p_1), \Delta t(p_2), \ldots, \Delta t(p_n)), \ n = |P|,$$

where $\Delta t(p) = -W(p, t) + W(t, p)$. For a sequence $w = t_1 t_2 \ldots t_n \in Seq(\mu_0)$ and $p \in P$, $\Delta w = \sum_{i=1}^{n} \Delta t_i$ and $\Delta w(p)$ is a p-th component of a vector Δw, i.e., $\Delta w(p) = \sum_{i=1}^{n} \Delta t_i(p)$. Note that if $\delta(\mu_0, w) = \mu_1$, $w \in Seq(\mu_0)$, then $\mu_1 = \mu_0 + \Delta w$ as a vector.

2 Some codes related to Petri nets

For a Petri net $PN = (P, A, W, \mu)$ and a subset $X \subseteq Re(\mu)$ we can define a deterministic automaton $A(PN)$ as follows: $Re(\mu)$, A, $\delta : Re(\mu) \times A \to Re(\mu)$, μ, and X, are regarded as the state set, the input set, the next-state function, the initial state, and the final set of $A(PN)$, respectively (For basic concepts of automata, refer to [1,p.10]). By using such automata, in [9] we defined four kinds of prefix codes and examined fundamental properties of these codes.
The set

$$Stab(PN) = \{w \mid w \in Seq(\mu) \text{ and } \delta(\mu, w) = \mu\}$$

forms a submonoid of A^*. If $Stab(PN) \neq \{1\}$, then we denote the base of $Stab(PN)$ by $S(PN)$. Since $S(PN)A^+ \cap S(PN) = \emptyset$, $S(PN)$ is a prefix code over A.

A submonoid M of A^* is called *pure* [6] if for all $x \in A^*$ and $n \geq 1$,

$$x^n \in M \Longrightarrow x \in M.$$

A subsemigroup H of a semigroup S is *extractable* in S [8, p.191] if

$$x, y \in S,\ z \in H,\ xzy \in H \Longrightarrow xy \in H.$$

Proposition 2.1. $Stab(PN)$ is an extractable pure monoid.

Proof. It is clear that $Stab(PN)$ is right unitary. Let $y, xy \in Stab(PN)$. Then x is a sequence from the initial marking μ. Since $\Delta y = 0$(the zero vector) and $\Delta(xy) = \Delta x + \Delta y = 0$, we have $x \in Stab(PN)$. Thus $Stab(PN)$ is left unitary. Therefore $Stab(PN)$ is biunitary.

Assume that $x^n \in Stab(PN), n \geq 1$. Then it is obvious that x is a squence from μ. Since $\Delta(x^n)\Delta x = 0$, we have $\Delta x = 0$DThus $x \in Stab(PN)$, and $Stab(PN)$ is pure.

Let $x, y \in A^*$ and $z, xzy \in Stab(PN)$. If $x = 1$, then $z, zy \in Stab(PN)$. Since $Stab(PN)$ is biunitary, we have $y \in Stab(PN)$ and $xy \in Stab(PN)$. Similarly $y = 1$ implies $xy \in Stab(PN)$. Suppose that $x, y \in A^+$. y is a sequence from $\mu + \Delta xz = \mu + \Delta x$. Thus xy is a squence from μ. From $\mu + \Delta(xzy) = \mu + \Delta(xy) = \mu$ we have $xy \in Stab(PN)$. □

Definition 2.1. Let $PN = (P, A, W, \mu)$ be a Petri net with a positive marking μ. Define the subset $D(PN)$ as the set of all positive sequence w of $S(PN)$.

Since $D(PN)$ is a subset of a bifix code $S(PN)$, also $D(PN)$ is a bifix code over A if $D(PN) \neq \emptyset$. By the same argument mentioned above, we have

Proposition 2.2. If $D(PN) \neq \emptyset$, then $D(PN)^*$ is an extractable pure monoid.

Example 2.1. Let $PN = (\{p, q\}, \{a, b\}, W, \mu)$ be a Petri net defined by $W(a, p) = W(p, b) = W(q, a) = W(b, q) = 1$, $\mu(p) = \mu(q) = 2$. Then $D(PN) =$

$\{ab, ba\}$, therefore $\{ab, ba\}^*$ is pure [1, p.324].

Proposition 2.3. If $x, y \in A^+$, z, $xzy \in D(PN)$, then $xz^*y \subseteq D(PN)$.

Proof Let $x, y \in A^+$, $z, xzy \in D(PN)$ and μ an initial marking of PN.

First we show that $xy \in D(PN)$. x is a positive sequence from μ, and y is a positive sequence from $\mu + \Delta(xz) = \mu + \Delta x$. Therefore $xy \in PSeq(\mu)$. Since $\Delta(xzy) = \Delta(xy) = 0$, we have that $xy \in D(PN)^*$, so that $xy = d_1 \cdots d_m$ for some $d_i \in D(PN)$, $m \geq 1$. We have the following three cases.
Case (a). $m = 1, xy = d_1 \in D(PN)$
Case (b). $m \geq 2$, $x = d_1 u$ for some $u \in A^*$.
Case (c). $m \geq 2$, $d_1 = xu, y = uv, v = d_2 \cdots d_m$ for some $u, v \in A^*$.
In Case (b) we have $d_1, xzy = d_1 uzy \in D(PN)$, but it dose not occur since $D(PN)$ is a prefix code. In Case (c), if $u = 1$, then $d_1, xzy = d_1 zy \in D(PN)$DThis contradicts the fact that $D(PN)$ is a prefix code. Therefore $u \neq 1$DIt follows that both d_m and $xzy = xzud_2 \cdots d_m$ are elements of $D(PN)$. This contradicts the fact that $D(PN)$ is a suffix code. Therefore only Case (a) is possible to occur. Thus $xy \in D(PN)$.

Next we show that $x, y \in A^+$, z, $xzy \in D(PN)$ implies $xz^2y \in D(PN)$. Since z is a positive sequence from $\mu + \Delta x = \mu + \Delta(xz)$, we have $xz^2 \in PSeq(\mu)$. Since y is a positive sequence from $\mu + \Delta(xz) = \mu + \Delta(xz^2)$, we have $xz^2y \in PSeq(\mu)$. Therefore, from $\Delta(xz^2y) = \Delta(xzy) = 0$ we have $xz^2y \in D(PN)^*$DThus

$$xz^2y = d_1 \cdots d_m,\ d_i \in D(PN),\ m \geq 1.$$

We have the following four cases.
Case 1. $m = 1$, $xz^2y = d_1 \in D(PN)$,
Case 2. $m \geq 2$, $d_1 = xz^2u$, $y = ud_2 \cdots d_m$ for some $u \in A^*$,
Case 3. $m \geq 2$, $d_1 = xzu$, $z = uv$, $vy = d_2 \cdots d_m$ for some $u, v \in A^*$,
Case 4. $m \geq 2$, $d_1 = xu$, $z = uv$, $vzy = d_2 \cdots d_m$, $u, v \in A^*$ for some $u, v \in A^*$,
Case 5. $m \geq 2$, $x = d_1 u$, for some $u \in A^*$.

In Case 2, $d_m, xzy = xzud_2 \cdots d_m \in D(PN)$. Thus Case 2 cannot occur since $D(PN)$ is a suffix code. In Case 3, $d_m \in D(PN)$, and $xzy = xuvy = xud_2 \cdots d_m \in D(PN)$. However Case 3 cannot occur since $D(PN)$ is a suffix code. Since $D(PN)$ is a prefix code, Case 4 and Case 5 cannot occur. Therefore only Case 1 is possible to occur. Thus $xz^2y \in D(PN)$.

Now suppose that $x, y \in A^+, z, xz^ny \in D(PN)$, $n \geq 2$. Then, xz^{n-1}, $y \in A^+$, z, $(xz^{n-1})zy \in D(PN)$. Therefore we have $(xz^{n-1})z^2y = xz^{n+1}y \in D(PN)$. □

Let C be a code over A. C is an *infix* code ([7,p.129]), if for all $x, y, z \in A^*$,

$$z,\ xzy \in C \Longrightarrow x = y = 1,$$

Proposition 2.4. If $D(PN)$ is a non-empty finite set, then $D(PN)$ is an infix code.

Proof Let $x, y \in A^*$, $z, xzy \in D(PN)$. $x = 1, y \neq 1$ or $x \neq 1, y = 1$ cannot ocuur because $D(PN)$ is a bifix code. Therefore either $x = y = 1$ or $x, y \in A^+$. By Proposition 2.3, $x, y \in A^+$ and z, $xzy \in D(PN)$ follow that $xz^*y \in D(PN)$. This contradicts the fact that $D(PN)$ is a finite set. Thus we have $x = y = 1$. □

Example 2.2. (1). Let $PN=(\{p,q,r\},\{a,b,c,d\},W,\mu_0)$ be a Petri net such that $W(p,a) = W(a,q) = W(q,b) = W(b,r) = W(r,c) = W(c,q) = W(q,d) = W(d,p) = 1$, $\mu_0 = (2,1,1)$. Then $D(PN) = a(bc)^*d$. Therefore the infinite code $D(PN)$ is infix. Thus the converse of Proposition 2.4 is false.
(2). Let $PN = (\{p\},\{a,b\},W,\mu_0)$ be a Petri net such that $W(a,p) = 1, W(p,b) = 1$, $\mu_0 = (1)$. Then $ab, a^2b^2 \in D(PN)$. Therefore the infinite code $D(PN)$ is not infix.

3 Limited code

A submonoid M of A^* is *very pure* if for all $u, v \in A^*$,

$$uv,\ vu \in M \Rightarrow u,\ v \in M.$$

The base of a very pure monoid is called a *circular* code.

Let $p, q \geq 0$ be two integers. A code C is called $(p,\ q)$-*limited* if for any sequence $u_0, u_1, ..., u_{p+q}$ of words in A^*, the assumptions $u_{i-1}u_i \in C^*$ $(1 \leq i \leq p+q)$ imply $u_p \in C^*$.

Any limited code is circular ([1, p.329, Proposition 2.1]). If a subset C of A^* is a bifix (1,1)-limited code, then for any $u_0, u_1, u_2 \in A^*$ such that $u_0u_1, u_1u_2 \in C^*$ we have $u_1 \in C^*$. Thus $u_0u_1, u_1, u_1u_2 \in C^*$. This imples that $u_0, u_1, u_2 \in C^*$ because C^* is biunitary. Therefore C is (p,q)-limited for all p, q with $p + q = 2$.

Let $PN_0 = (\{p\},\{a,b\},W,\mu_0)$ be a Petri net such that $W(a,p) = \alpha, W(p,b) = \beta$, $\mu_0 = (\lambda_p)$, $\lambda_p > 0$.

Consider the set Ω of all positive markings in PN_0;

$$\Omega = \{\mu \mid \mu = \mu_0 + \Delta w,\ w \in PSeq(\mu_0)\}.$$

Let $g = gcd(\alpha, \beta)$ be the greatest common divisor of α and β, and let $\mathbb{N} = \{0, 1, 2, \cdots\}$ be the set of non-negative integers. Then we have
(0) $D(PN_0)$ is dense, that is, $D(PN_0) \cap A^*wA^* \neq \emptyset$ for every $w \in A^*$.
(1) If $\lambda_p < g$, then $\Omega = \{\lambda_p + ng \mid n \in \mathbb{N}\}$.
(2) If $\lambda_p = sg$, $s \geq 1$, $s \in \mathbb{N}$, then $\Omega = \{ng \mid n \geq 1, n \in \mathbb{N}\}$.
(3) If $\lambda_p = sg + t_p$, $s \geq 0$, $0 < t_p < g$, then $\Omega = \{t_p + ng \mid n \geq 0,\ n \in \mathbb{N}\}$.

Proposition 3.1. If $\lambda_p > gcd(\alpha, \beta)$, then $D(PN_0)$ is not circular.

Proof. Let $D = D(PN_0)$, and let $g = gcd(\alpha, \beta)$. Note that $\mu_0 = \lambda_p$. We have the following two cases:
Case 1. $g = \alpha$ or $g = \beta$. Case 2. $\alpha = \alpha' g$, $\beta = \beta' g$, $\alpha' \geq 2$, $\beta' \geq 2$, $gcd(\alpha', \beta') = 1$.

Case 1-(i). If $\alpha = gcd(\alpha, \beta), \beta = k\alpha, k > 1$, then $aa^{k-1}b$, $a^{k-1}ba \in D$ and $a \notin D^*$. Therefore D is not circular.

Case 1-(ii). If $\beta = gcd(\alpha, \beta)$, $\alpha = k\beta$, $k > 1$, then $ab^{k-1}b$, $bab^{k-1} \in D$ and $b \notin D^*$. Thus D is not circular.

Case 1-(iii). If $\alpha = gcd(\alpha, \beta), \alpha = \beta$, then $ab, ba \in D$. Consequently D is not circular.

Case 2. Since $g = gcd(\alpha, \beta)$, there exist some integers x' and y' such that $\alpha x' + \beta y' = g$.

Case 2-(i). We consider the case $\alpha x' + \beta y' = g, x' > 0, y' < 0$. We set $x = x', y = -y'$, then $\alpha x - \beta y = g, x > 0, y > 0$. Since $\alpha x = \beta y + g > \beta i$, for $i = 1, \cdots, y$, b^y is a sequence from $\lambda_p + \Delta(a^x)$, and $\lambda_p + \Delta(a^x b^y) = \lambda_p + g$. Consequently $a^x b^y$ is also a sequence from $\lambda_p + \Delta(a^x b^y)$, therefore $(a^x b^y)^2 \in PSeq(\mu_0)$. Similarly we have $(a^x b^y)^{\beta'} \in PSeq(\mu_0)$ and $\lambda_p + \Delta((a^x b^y)^{\beta'}) = \lambda_p + \beta' g$. Thus $(a^x b^y)^{\beta'} b \in D(PN_0)$. On the other hand, since $\lambda_p > g$, we have $\lambda_p + \Delta((a^x b^y)^{\beta'-1} b) = \lambda_p - g$. It follows that $(a^x b^y)^{\beta'-1} b \cdot a^x b^y \in D$. However $a^x b^y \notin D^*$. Thus D is not circular.

Case 2-(ii). We consider the case $-\alpha x + \beta y = g$ for some positive integers x and y. Then $a(a^x b^y)^j \in PSeq(\mu_0)$, $1 \leq j \leq \alpha'$. Thus $a(a^x b^y)^{\alpha'} \in D$. On the other hand, from $\lambda_p > g$ and $\alpha' \geq 2$ we have $\lambda_p + \Delta(a^x b^y) = \lambda_p - g > 0$. Thus $a^x b^y a \in PSeq(\mu_0)$. It follows that $a^x b^y a (a^x b^y)^{\alpha'-1} \in D$. However $a^x b^y \notin D^*$. Therefore D is not circular. □

Proposition 3.2. If $\lambda_p \leq gcd(\alpha, \beta)$, then any nonempty subset of $D(PN_0)$ is (p, q)-limited for all p, q with $p + q = 2$.

Proof. Let $D = D(PN_0)$ and $g = gcd(\alpha, \beta)$. Let d be an arbitrary element in D. Then d has a proper right factor $v \neq 1, d$, because $a, b \notin D$. Let $d = uv$, $u, v \in A^+\mathrm{D}$

First, we shall show that $\Delta v \leq -g$. Assume the contrary. Then $\Delta v > -g$, and we have $\Delta v \geq 0$ since Δv is a multiple of g. If $\Delta v = 0$, then $\Delta u = 0$ since $\Delta d = \Delta(uv) = \Delta u + \Delta v = 0$. Therefore we get $u \in D^*$. This contradicts the fact that D is a prefix code. Thus we have $\Delta v > 0$, it follows that $\Delta v \geq g$ and $\Delta u = -\Delta(v) \leq -g$. Then we have $\mu_0 + \Delta u = \lambda_p + \Delta u \leq \lambda_p - g \leq 0$, showing that $u \notin PSeq(\mu_0)$ and contradicting $d \in D$. Therefore we have prove $\Delta v \leq -g$.

Next we shall show that any nonempty subset C of D is (1,1)-limited. Note that C is a bifix code. Suppose that $u_0, u_1, u_2 \in A^*$ and $u_0 u_1$, $u_1 u_2 \in C^*$. If $u_i = 1$ for some i, $0 \leq i \leq 2$, then $u_0, u_1, u_2 \in D^*$ since C^* is biunitary. We assume that $u_i \neq 1$ for all $i, 0 \leq i \leq 2$. We may write

$$u_0 = v_0 x_1,\ u_1 = y_1 w_1,\ x_1 y_1 \in C,\ v_0,\ w_1 \in C^*.$$

If $y_1 \neq 1$ and $y_1 \notin C$, then y_1 is a proper right factor of $d_i \in D$. Therefore $\Delta(y_1) \leq -g$ as mentioned above. It follows $\lambda_p + \Delta y_1 \leq 0$, and $y_1 \notin PSeq(\mu_0)$. However, $u_1u_2 = y_1w_1u_2 \in C^* \subseteq D^*$. Thus $y_1 \in PSeq(\mu_0)$. This is a contradiction. Therefore $y_1 = 1$ or $y_1 \in C$. Thus $u_1 \in C^*$. □

Let $PN_1 = (\{p,q\}, \{a,b\}, W, \mu_0)$ be a Petri net such that $W(a,p) = \alpha > 0, W(p,b) = \alpha' > 0$, $W(q,a) = \beta > 0$, $W(b,q) = \beta' > 0$, $\mu_0(p) = \lambda_p$, $\mu_0(q) = \lambda_q$. We examine the code $D(PN_1)$ associated with Petri net PN_1.

Suppose that $D(PN_1) \neq \emptyset$ and $w \in D(PN_1)$. Let $n = |w|_a$ and $m = |w|_b$, then $\Delta(w)\Delta(a) + m\Delta(b) = 0$. Consequently the linear equation

$$\begin{pmatrix} \alpha & -\alpha' \\ -\beta & \beta' \end{pmatrix} \begin{pmatrix} n \\ m \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

has a non-trivial solution. Thus $\alpha\beta' = \alpha'\beta$. Therefore, if $D(PN_1) \neq \emptyset$, then $PN_1 = (\{p,q\}, \{a,b\}, W, \mu_0)$ has the following form:
$W(a,p) = \alpha, W(p,b) = k\alpha, W(q,a) = \beta, W(b,q) = k\beta$, for some $k > 0$.
Here we assume that k is a positive integer. That is, we define a Petri net $PN_1 = (\{p,q\}, \{a,b\}, W, \mu_0)$ as follows:

$$\Delta(a) = \begin{pmatrix} \alpha \\ -\beta \end{pmatrix}, \quad \Delta(b) = \begin{pmatrix} -k\alpha \\ k\beta \end{pmatrix},$$

where k is a positive integer.

We define an integer M_p as follows

$$M_p = \begin{cases} \dfrac{\lambda_p}{\alpha} - 1, & \text{if } \dfrac{\lambda_p}{\alpha} \text{ is an integer,} \\ \left[\dfrac{\lambda_p}{\alpha}\right], & \text{if } \dfrac{\lambda_p}{\alpha} \text{ is not an integer,} \end{cases}$$

where [ ] is the symbol of Gauss. Similarly we difine an integer M_q as follows, $M_q = \frac{\lambda_q}{\beta} - 1$ if $\frac{\lambda_q}{\beta}$ is an integer, and $M_q = [\frac{\lambda_p}{\beta}]$ if $\frac{\lambda_q}{\beta}$ is not an integer. Note that $a, a^2, \cdots, a^{M_q} \in PSeq(\mu_0)$ and $a^{M_q+1} \notin PSeq(\mu_0)$. If $M_p \geq k$, then $b \in PSeq(\mu_0)$.

Now, we observe that $M_p + M_q \leq k-1$ implies $D(PN_1) = \emptyset$. If $M_p + M_q \leq k-1$, then $M_p \leq k-1$. It follows that $b \notin PSeq(\mu_0)$. Furthermore, if $M_q = 0$, then $a \notin PSeq(\mu_0)$ and $PSeq(\mu_0) = \{1\}$. If $M_q > 0$, we have

$$a^i \in PSeq(\mu_0),\ 1 \leq i \leq M_q, \text{ and } \Delta a^i(p) = \lambda_p + \alpha i \leq \lambda_p + \alpha M_q.$$

Assume that $\lambda_p + \alpha M_q - \alpha k > 0$. If $\frac{\lambda_p}{\alpha}$ is an integer, then $M_p + 1 + M_q - k > 0$. This contradicts the hypothesis. If $\lambda_p = \alpha M_p + s$ for some s, $1 \leq s < \alpha$, then

$$0 < M_p + M_q + \frac{s}{\alpha} - k < M_p + M_q + 1 - k.$$

This also contradicts our hypothesis. Therefore we get $\lambda_p + \alpha M_q - \alpha k \leq 0$, showing that b is not enabled in $\mu_0 + \Delta(a^i)$, $1 \leq i \leq M_q$. Therefore $PSeq(\mu_0) = \{1, a, a^2, \cdots, a^{M_q}\}$. Thus the condition $M_p + M_q \leq k - 1$ implies $D(PN_1) = \emptyset$.

Lemma 3.3. Let $d = uv \in D(PN_1), u, v \in A^+$.
(1) If $M_p = 0$ and $M_q \geq k$, then $\Delta v(p) \leq -\alpha$.
(2) If $M_p \geq k$ and $M_q = 0$, then $\Delta v(q) \leq -\beta$.

Proof. (1). Suppose that $\Delta v(p) > -\alpha$. Then, since $\Delta v(p)$ is a multiple of α, we have $\Delta v(p) \geq 0$. Note that

$$\Delta v = |v|_a \Delta a + |v|_b \Delta b = \begin{pmatrix} (|v|_a - k|v|_b)\alpha \\ -(|v|_a - k|v|_b)\beta \end{pmatrix}.$$

If $\Delta v(p) = 0$, then $|v|_a - k|v|_b = 0$. Thus $\Delta v = 0$, it follows that $\Delta u = 0$ and $u \in D(PN_1)^*$. This contradicts the fact that $D(PN_1)$ is a prefix code. Thus $\Delta v(p) \geq \alpha$. It implies that $\Delta u(p) = -\Delta v(p) \leq -\alpha$. Since $M_p = 0$, $\mu_0(p) + \Delta u(p) < \lambda_p - \alpha \leq 0$. This yields $u \notin PSeq(\mu_0)$. This is a contradiction. Therefore we have $\Delta v(p) \leq -\alpha$.
(2). Proof is omitted. □

Proposition 3.4. We have
(1) If $M_p + M_q > k, M_p \geq k$ and $M_q \geq 1$, then $D(PN_1)$ is not circular.
(2) If $M_p + M_q > k, k > M_p \geq 1, M_q > 1$, then $D(PN_1)$ is not circular.
(3) If $M_p + M_q = k$, then $D(PN_1)$ is a singleton.
(4) If $M_p = 0, M_q > k$, then any nonempty subset of $D(PN_1)$ is (p, q)-limited for all p, q with $p + q = 2$.
(5) If $M_p > k, M_q = 0$, then any nonempty subset of $D(PN_1)$ is (p, q)-limited for all p, q with $p + q = 2$.

Proof. Let $D = D(PN_1)$.
(1) From $M_p \geq k$ it follows that $b \in PSeq(\mu_0)$ and $ba^k \in D$. On the other hand, from $\lambda_p > k\alpha$ we have $\lambda_p + \alpha - k\alpha > \alpha$. Thus $ab \in PSeq(\mu_0)$. Since $\lambda_p + (k-1)\beta > (k-1)\beta$, we have $aba^{k-1} \in D$. Therefore D is not circular.
(2) Let $k = M_p + r$DSince $M_p + M_q > k$, we have $M_q > r$. It follows that $a^r \in PSeq(\mu_0)$. Since $\lambda_p + r\alpha - k\alpha = \lambda_p - \alpha M_p > 0$, we have $a^r b \in PSeq(\mu_0)$. Consequently $a^r b a^{M_p} \in D$. On the other hand we have $a^{r+1} b a^{M_p - 1} \in D$ since $M_q > r$. Therefore D is not circular.
(3) First we consider the case that $M_p \geq 1, M_q \geq 1$. Since $M_p = k - M_q < k$, we have $b \notin PSeq(\mu_0)$. It is obvious that $a^i \in PSeq(\mu_0)$ for $i = 0, 1, \cdots, M_q$. If $i \leq M_q - 1$, then from $\lambda_p + i\alpha - k\alpha = \lambda_p - M_p\alpha - (M_q - i)\alpha$ and $\lambda_p - M_p\alpha \leq \alpha$, we get $\lambda_p + i\alpha - k\alpha \leq 0$, showing that $a^i b \notin PSeq(\mu_0)$, $0 \leq i \leq M_q - 1$. It is obvious that $a^{M_q} b \in PSeq(\mu_0)$ and

$$\mu_0 + \Delta(a^{M_q} b) = \begin{pmatrix} \lambda_p - \alpha M_p \\ \lambda_q + \beta M_p \end{pmatrix}.$$

For $j = 0, \cdots, k - M_q$, we have $a^{M_q}ba^j \in PSeq(\mu_0)$ since $\lambda_q + \beta M_p - \beta j \geq \lambda_q$. However $a^{M_q}ba^jb \notin PSeq(\mu_0)$ since

$$\lambda_p - \alpha M_p + \alpha j \leq \lambda_p - \alpha M_p + \alpha(k - M_q) = \lambda_p \leq k\alpha.$$

Therefore $D = \{a^{M_q}ba^{k-M_q}\}$. Similarly, if $M_p = 0, M_q = k$, then $D = \{a^kb\}$. If $M_p = k, M_q = 0$, then $D = \{ba^k\}$.
(4) Let C be a nonempty subset of D. Suppose that $w_0, w_1, w_2 \in A^*$ and w_0w_1, $w_1w_2 \in C^*$. If $w_i = 1$ for some i, $0 \leq i \leq 2$, then $w_0, w_1, w_2 \in C^*$ since C^* is biunitary. We assume that $w_i \neq 1$ for all $i, 0 \leq i \leq 2$. We may write

$$w_0 = u_0x_1,\ w_1 = y_1v_1,\ x_1y_1 \in C \text{ for some } x_1, y_1 \in A^*,\ u_0,\ v_1 \in C^*.$$

If either $y_1 = 1$ or $y_1 \in C$, then $w_1 \in C^*$. Assume that $y_1 \neq 1$ and $y_1 \notin C$. Then y_1 is a proper right factor of an element in D. Since $M_p \leq \alpha$, $\lambda_p + \Delta y_1 \leq 0$ by Lemma 3.3, we have $y_1 \notin PSeq(\mu_0)$. However $w_1w_2 = y_1v_1w_2 \in C^* \subset D^*$. Thus $y_1 \in PSeq(\mu_0)$. This is a contradiction. Therefore $y_1 \in C \cup \{1\}$. This yields $w_1 \in C^*$.
(5) The proof of (5) is similar to the proof of (4), therefore it is omitted. □

Remark 3.1. In the above proof for (3) we have $D = \{w\}$, $w = a^{M_q}ba^{k-M_q}$, $M_q \neq 0$, $k - M_q \neq 0$. D is (s, t)-limited for all $s, t \geq 0$ with $s + t = 3$. For any $n, m \geq 0$ the code $D = \{a^nba^{k-n}\} = \{a^nba^m\}, k + m$, is realizable as a Petri net code which is produced by the Petri net PN_1 such that $W(a,p) = W(q,a) = 1$, $W(p,b) = W(b,q) = k$, $\mu_0(p) = m + 1$, $\mu_0(q) + 1$.

Let $PN = (P, A, W, \mu_0)$ be a Petri net. By $PRe(\mu_0)$ we denote the set of all possible positive markings reachable from μ_0. For a Petri net PN we define a deterministic automaton $A(PN)$ as follows:
$PRe(\mu_0)$, A, $\delta : PRe(\mu_0) \times A \to PRe(\mu_0)$, μ_0, and $\{\mu_0\}$, are regarded as the state set, the input set, the next-state function, the initial state, and the final set of $A(PN)$, respectively.

Corollary 3.5. Let n and k be arbitrary integers such that $n > k > 1$. Define the automaton

$$\mathcal{A}_{(n,k)} = (\{1, 2, \cdots, n\}, \{a, b\}, f, 1, \{1\})$$

by $f(i,a) = i+1$, $1 \leq i \leq n-1$, $f(j,b) = j-k$, $k+1 \leq j \leq n$. Then any nonempty subset of the base of language $L(\mathcal{A}_{(n,k)})$ recognized by $\mathcal{A}_{(n,k)}$ is a (p,q)-limited code for all p, q with $p + q = 2$.

Proof. We define the $PN_1 = (\{p, q\}, \{a, b\}, W, \mu_0)$ as follows:
$W(a,p) = 1, W(p,b) = k, W(b,q) = k, W(q,a) = 1$, $\mu_0(p) = 1, \mu_0(q)$. Then $M_p = 0$, $M_q - 1 \geq k$. Therefore, by Proposition 3.4, $D(PN)$ is (1,1)-limited. Since $A(PN_1)$ is isomorphic to $\mathcal{A}_{(n,k)}$ as an automaton, we have Corollary 3.5. □

Proposition 3.6. Let $PN = (\{p_1, \cdots, p_n\}, \{a_1, \cdots, a_n\}, W, \mu_0), n \geq 2$, be a Petri net such that $W(p_i, a_i) = \alpha_i, W(a_i, p_{i+1}) = \beta_i, 1 \leq i \leq n-1$, and $W(p_n, a_n) = \alpha_n, W(a_n, p_1) = \beta_n$, $\mu_0 = (\lambda_1, \cdots, \lambda_n), \mu_0(p_i) = \lambda_i$, $1 \leq i \leq n$. Furthermore let $g_j = gcd(\beta_{j-1}, \alpha_j), 2 \leq j \leq n$. If $\lambda_1/\alpha_1 > 1$ and $\lambda_i \leq g_i$ for all $i = 2, \cdots, n$, and if $D(PN) \neq \emptyset$, then any nonempty subset of $D(PN)$ is (p, q)-limited for all p, q with $p + q = 2$.

Proof. We set $D = D(PN)$. Since $\lambda_i \leq g_i$ for all $i = 2, \cdots, n$, we have $D \subset a_1 A^+$. Let $d \in D$, $d = auv, u \in A^*, v \in A^+$DNote that v is a proper right factor of an element in D.

First we show that $\Delta v(p_i) \leq 0$ for all $i = 2, \cdots, n$. Suppose that $\Delta v(p_j) > 0$ for some $j \geq 2$. Since $\Delta v(p_j) > 0$ is a linear combination of β_{j-1} and α_j, $\Delta v(p_j)$ is a multiple of g_j. Therefore $\Delta v(p_j) > 0$ implies $\Delta(v)(p_j) \geq g_j$. Thus $-\Delta v(p_j) \leq -g_j$. On the other hand, $\Delta d = \Delta(a_1 u) + \Delta v = 0$, and we have $\Delta(a_1 u) = -\Delta v$. Therefore $\Delta(a_1 u)(p_j) = -\Delta v(p_j) \leq -g_j$. However, $\mu_0(p_j) + \Delta(a_1 u)(p_j) \leq \lambda_j - g_j \leq 0$. This contradicts the fact that $a_1 u \in PSeq(\mu_0)$. Consequently we have that $\Delta v(p_i) \leq 0$ for all i, $(i \geq 2)$.

Next we show that $v \notin PSeq(\mu_0)$. To prove this we show that there exists some p_t, $t \geq 2$, such that $\Delta v(p_t) \leq -g_t$. Suppose the contrary. Then $\Delta v(p_i) = 0$ for all $i \geq 2$. Let x_j be the number of the letter a_j in v, then

$$\Delta(v) = \begin{pmatrix} -\alpha_1 & 0 & \cdots & 0 & \beta_n \\ \beta_1 & -\alpha_2 & \cdots & 0 & 0 \\ 0 & \beta_2 & \cdots & 0 & 0 \\ & & \cdots & & \\ 0 & 0 & \cdots & \beta_{n-1} & -\alpha_n \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ x_n \end{pmatrix} = \begin{pmatrix} \Delta(v)(p_1) \\ 0 \\ \cdot \\ \cdot \\ 0 \end{pmatrix}.$$

We regard the equation above as a system of linear equations. Since $D \neq \emptyset$, the determinant of a matrix $(\Delta a_1, \cdots, \Delta a_n)$ is zero. That is, $\alpha_1 \alpha_2 \cdots \alpha_n = \beta_1 \beta_2 \cdots \beta_n$. Since there exists a solution, we must have $\Delta v(p_1) = 0$. Consequently $\Delta(a_1 u) = -\Delta v = 0$ and $a_1 u \in PSeq(\mu_0)$. Therefore $a_1 u \in D^*$. This contradicts the fact that D is a prefix code. Thus we have proved that $\Delta v(p_t) \leq -g_t$ for some p_t, $t \geq 2$. This means that $v \notin PSeq(\mu_0)$ since $\mu_0(p_t) + \Delta(v)(p_t) \leq \lambda_j - g_t \leq 0$.

Finally we prove that any nonempty subset C of D is (1,1)-limited. Suppose that $w_0, w_1, w_2 \in A^*$, and $w_0 w_1, w_1 w_2 \in C^*$. We may write

$$w_0 = u_0 x_1,\ w_1 = y_1 v_1,\ x_1 y_1 \in C \text{ for some } x_1, y_1 \in A^*,\ u_0, v_1 \in C^*.$$

Note that y_1 is a right factor of an element of D. If $y_1 \neq 1$ and $y_1 \notin C$, then $y_1 \notin PSeq(\mu_0)$ as we mentioned above. Therefore $w_1 w_2 = y_1 v_1 w_2 \notin PSeq(\mu_0)$. This contradicts our hypothesis. Thus $y_1 = 1$ or $y_1 \in C$. This shows that $w_1 \in C^*$. □

Let $PN_2 = (\{p_1, p_2\}, \{a, b, c\}, W, \mu_0)$ be a Petri net such that $W(a, p_1) = \alpha_1, W(p_1, b) = \alpha_2$, $W(b, p_2) = \beta_1, W(p_1, c) = \alpha_3, W(p_2, c) = \beta_2$, $\mu_0(p_1) = \lambda_1$, $\mu_0(p_2) = \lambda_2$.

Lemma 3.7. Let PN_2 be a Petri net mentioned above, and let $\alpha = gcd(\alpha_1, \alpha_2, \alpha_3)$, $\beta = gcd(\beta_1, \beta_2)$. Suppose that $D(PN_2) \neq \emptyset$ and $\lambda_1 \leq \alpha$, $\lambda_2 \leq \beta$. If $d \in D(PN_2)$ and v, $v \neq 1$, is a proper right factor of d, then we have one of the following:
(1) $\Delta v(p_1) \leq -\alpha$, $\Delta v(p_2) \leq -\beta$.
(2) $\Delta v(p_1) = 0$, $\Delta v(p_2) \leq -\beta$.
(3) $\Delta v(p_1) \leq -\alpha$, $\Delta v(p_2) = 0$.

Proof. Let $D = D(PN_2)$. It is obvious that b and c are not enabled in μ_0, so that $D \subset aA^*$. Let $d \in D$, $d = auv, u \in A^*, v \in A^+$. If $u = 1$, then $\Delta v = -\Delta a$ and (3) holds. Assume $u \neq 1$. If $\Delta v(p_1) > 0$, then $\Delta v(p_1) \geq \alpha$ because $\Delta v(p_1)$ is a multiple of α. Thus $\Delta(au)(p_1) = -\Delta v(p_1) \leq -\alpha$. Hence $\mu_0(p_1) + \Delta(au)(p_1) \leq \mu_0(p_1) - \alpha \leq 0$. This contradicts the fact that $au \in PSeq(\mu_0)$. Therefore $\Delta v(p_1) \leq 0$. Similarly we have $\Delta v(p_2) \leq 0$. If $\Delta v(p_1) = 0$ and $\Delta v(p_2) = 0$, i.e., $\Delta v = 0$, then $\Delta(au) = 0$. Since $au \in PSeq(\mu_0)$, we have $au \in D^*$, contradicting the fact that D is a prefix code. Therefore at least one of (1),(2) or (3) occurs. □

Proposition 3.8. If $D(PN_2) \neq \emptyset$ and $\lambda_1 \leq \alpha$, $\lambda_2 \leq \beta$, then any nonempty subset of $D(PN_2)$ is (p, q)-limited for all p, q with $p + q = 2$.

Proof. Let $D = D(PN_2)$. We note that for a right factor v, $v \neq 1$, of an element $d \in D$ the vector $\mu_0 + \Delta v$ is not positive by Lemma 3.7. That is, $v \notin PSeq(\mu_0)$. Let C be a nonempty subset of D. Assume $w_0, w_1, w_2 \in A^*$ and $w_0 w_1, w_1 w_2 \in C^*$. If either $w_0 = 1$ or $w_1 = 1$, then $w_1 \in C^*$. Therefore we consider the case where $w_0 \neq 1$ and $w_1 \neq 1$. Let $w_0 w_1 = d_1 \cdots d_m$, $d_i \in C, 1 \leq i \leq m$. There exist an integer i, $1 \leq i \leq m$, and u, $v \in A^*$ such that $w_0 = d_1 \cdots d_{i-1} u$, $d_i = uv$, $w_1 = v d_{i+1} \cdots d_m$. If $v \neq 1$ and $v \notin D$, then v is a proper right factor d_i. Thus $v \notin PSeq(\mu_0)$. However, from $w_1 w_2 = v d_{i+1} \cdots d_m w_2 \in C^*$, we heve $v \in PSeq(\mu_0)$. This is a contradiction. Hence $v = 1$ or $d \in C$. It follows that $w_1 \in C^*$. Thus C is (1,1)-limited. □

Let $PN_3 = (\{p, q\}, \{a, b, c\}, W, \mu_0)$ be a Petri net such that $W(a, p) = \alpha$, $W(q, a) = \beta, W(p, b) = \alpha + \beta, W(b, q) = \alpha + \beta, W(c, p) = \beta, W(q, c) = \alpha$, $\mu_0(p) = \lambda_p, \mu_0(q) = \lambda_q$.

Lemma 3.9. Let PN_3 be a Petri net mentioned above. If $\beta < \lambda_p \leq \alpha + \beta$ and $\beta < \lambda_q \leq \alpha$, then for any $u \in PSeq(\mu_0)$ we have one of the following:

$$(1)\ \Delta u = \begin{pmatrix} k(\alpha - \beta) \\ k(\alpha - \beta) \end{pmatrix},\ k \geq 0, \quad (2)\ \Delta u = \begin{pmatrix} k(\alpha - \beta) + l\alpha \\ k(\alpha - \beta) - l\beta \end{pmatrix},\ k \geq 0,\ l \geq 1,$$

$$(3)\ \Delta u = \begin{pmatrix} k(\alpha-\beta)-l\beta \\ k(\alpha-\beta)+l\alpha \end{pmatrix},\ k \geq 0,\ l \geq 1.$$

Proof. We shall prove the lemma by induction on the length of u in $PSeq(\mu_0)$. Since $\beta < \lambda_p \leq \alpha+\beta$, and $\beta < \lambda_q \leq \alpha$, only a is enabled in μ_0. That is, a positive sequence of length 1 is only a, and Δa is of the form (2).
Since $\lambda_q - \beta < \alpha - \beta < \alpha$, c is not enabled in $\mu_0 + \Delta a$. $\Delta(a^2)$ is of the form (2), and $\Delta(ab) = (-\beta, \alpha)$ is of the form (3). c is not enabled in $\mu_0 + \Delta(a^2)$. b is not enabled in $\mu_0 + \Delta(ab)$. $\Delta(a^3)$ is of form (2). $\Delta(a^2b)$, $\Delta(aba)$ and $\Delta(abc)$ are of the form (1).

Now we suppose that for $u \in PSeq(\mu_0), |u| > 3$, the vector Δu has a form (1),(2) or (3). Let x be an element in $\{a, b, c\}$ such that $ux \in PSeq(\mu_0)$. We shall show that $\Delta(ux)$ is of the form (1),(2) or(3).

Case 1. $\Delta u = (k(\alpha-\beta), k(\alpha-\beta))$. $\Delta(ua)$ is of the form (2). If $k = 0$, then both b and C are not enabled in $\mu_0 + \Delta u$. For $k \geq 1$, $\Delta(ub) = ((k-1)(\alpha-\beta) - 2\beta, (k-1)(\alpha-\beta) + 2\alpha)$ is of the form (3). $\Delta(uc) = ((k-1)(\alpha-\beta)+\alpha, (k-1)(\alpha-\beta)-\beta)$ is of the form (2).

Case 2. $\Delta u = (k(\alpha-\beta)+l\alpha, k(\alpha-\beta)-l\beta)$. $\Delta(ua)$ is of the form (2). If $l = 1$, then $\Delta(ub)$ is of the form (3). If $l \geq 2$, then $\Delta(ub) = ((k+1)(\alpha-\beta)+(l-2)\alpha, (k+1)(\alpha-\beta)-(l-2)\beta)$ is the form (1) or (2). If $k = 0$, then c is not enabled in $\mu_0 + \Delta u$. For $k \geq 1$, $\Delta(uc) = ((k-1)(\alpha-\beta)+2\alpha, (k-1)(\alpha-\beta)-2\beta)$.

Case 3. $\Delta u = (k(\alpha-\beta)-l\beta, k(\alpha-\beta)+l\alpha)$. $\Delta(ua) = ((k+1)(\alpha-\beta)-(l-1)\beta, (k+1)(\alpha-\beta)+(l-1)\alpha)$ is of the form (1) or (3). $\Delta(uc) = (k(\alpha-\beta)-(l-1)\beta, k(\alpha-\beta)+(l-1)\alpha)$. If $k = 0$, then b is not enabled in $\mu_0 + \Delta u$. For $k \geq 1$, $\Delta(ub) = ((k-1)(\alpha-\beta)-(l+2)\beta, (k-1)(\alpha-\beta)+(l+2)\alpha)$. Thus Lemma 3.9 is proved. □

Proposition 3.10. If $D(PN_3) \neq \emptyset$, and if $\beta < \lambda_p \leq \alpha+\beta$ and $\beta < \lambda_q \leq \alpha$, then any nonempty subset of $D(PN_3)$ is (p, q)-limited for all p, q with $p+q = 2$.

Proof. Let $D = D(PN_3)$, and let v, $v \neq 1$, be a proper right factor of $d \in D$.

First we shall show that $v \notin PSeq(\mu_0)$. Let $d = uv, u, v \in A^+$, then $\Delta v = -\Delta u$. Since $u \in PSeq(\mu_0)$, by Lemma 3.9 we have one of the following:

$$\text{(i)}\ \Delta v = \begin{pmatrix} -k(\alpha-\beta) \\ -k(\alpha-\beta) \end{pmatrix},\ k \geq 0, \qquad \text{(ii)}\ \Delta v = \begin{pmatrix} -k(\alpha-\beta)-l\alpha \\ -k(\alpha-\beta)+l\beta \end{pmatrix},\ k \geq 0,\ l \geq 1,$$

$$\text{(iii)}\ \Delta v = \begin{pmatrix} -k(\alpha-\beta)+l\beta \\ -k(\alpha-\beta)-l\alpha \end{pmatrix},\ k \geq 0,\ l \geq 1.$$

We consider Case (iii). If $v \in PSeq(\mu_0)$, then, by Lemma 3.9 we have the following three cases. Case (iii)-(1)

$$\Delta v = \begin{pmatrix} -k(\alpha-\beta)+l\beta \\ -k(\alpha-\beta)-l\alpha \end{pmatrix} = \begin{pmatrix} x(\alpha-\beta) \\ x(\alpha-\beta) \end{pmatrix},\ k \geq 0,\ l \geq 1,\ x \geq 0.$$

In this case, there is not a solution for xD
Case (iii)-(2)

$$\Delta v = \begin{pmatrix} -k(\alpha-\beta)+l\beta \\ -k(\alpha-\beta)-l\alpha \end{pmatrix} = \begin{pmatrix} x(\alpha-\beta)+y\alpha \\ x(\alpha-\beta)-y\beta \end{pmatrix},\ k \geq 0,\ l \geq 1,\ x \geq 0,\ y \geq 1.$$

In this case, only one solution of linear system is a non-positive $(x, y) = (-(k+l), l)$.
Case (iii)-(3)

$$\Delta v = \begin{pmatrix} -k(\alpha-\beta)+l\beta \\ -k(\alpha-\beta)-l\alpha \end{pmatrix} = \begin{pmatrix} x(\alpha-\beta)-y\beta \\ x(\alpha-\beta)+y\alpha \end{pmatrix},\ k \geq 0,\ l \geq 1,\ x \geq 0,\ y \geq 1.$$

In this case, only one solution of linear system is a non-positive $(x, y) = (-k, -l)$. Therefore in any cases we have $v \notin PSeq(\mu_0)$. Similarly, in Case (i) or Case (ii) we cannot write Δv in the form (1), (2) or (3) of Lemma 3.9. Therefore $v \notin PSeq(\mu_0)$.

Let C be a subset of D. Assume $w_0, w_1, w_2 \in A^*$ and $w_0w_1, w_1w_2 \in C^*$. If either $w_0 = 1$ or $w_1 = 1$, then $w_1 \in C^*$. Therefore we consider the case where $w_0 \neq 1$ and $w_1 \neq 1$. Let $w_0w_1 = d_1 \cdots d_m$, $d_j \in C, 1 \leq j \leq m$. There exist an integer i, $1 \leq i \leq m$, and $u, v \in A^*$ such that $w_0 = d_1 \cdots d_{i-1}u$, $d_i = uv$, $w_1 = vd_{i+1} \cdots d_m$. If $v \neq 1$ and $v \neq d_i$, then v is a proper right factor d_i. By using the above fact that $v \notin PSeq(\mu_0)$, we obtain $w_1 \notin PSeq(\mu_0)$. This is a contradiction. Thus we have either $v = 1$ or $v = d_i$ which implies $w_1 \in C^*$. Thus C is (1,1)-limited. □

When a submonoid of a free monoid is given, it seems complicated to judge whether the submonoid is pure or not. This is because we have to show it by the treatment of many different cases of words which belong to the submonoid. Also it doesn't seem easy to decide whether the base of a pure monoid is limited or not. Proposition 2.1 and 2.2 ensure that any submonoid generated by a code $D(PN)$ or $S(PN)$ is always pure. The proof techniques of Proposition 3.2-3.10 which use the properties of right factors of the elements in $D(PN)$ may be usable to decide whether $D(PN)$ is limited or not in other Petri nets.

References

[1] Berstel, J., and Perrin, D. Theory of Codes. *Academic Press*, 1985.

[2] Ito, M., and Kunimochi,Y. Some Petri Net Languages and Codes. *Lecture Note in Computer Science*, 2295:69–80, 2002.

[3] Lallement, G. Semigroup and Combinatorial Applications. *Wiley*, 1979.

[4] Murata, T. Petri Nets: Properties, Analysis and Applications. *Proceeding of the IEEE*, 77(4):541–580, 1989.

[5] Peterson, J. L. Petri Net Theory and the Modeling of Systems. *Prentice Hall*, 1981.

[6] Restivo, A. On a Question of McNaughton and Papert. *Information and Control*, 25:93–101, 1974.

[7] Shyr, H. J. Free monoids and languages. *Hon Min Book Company, Taichung, Taiwan*, 1991.

[8] Tamura, T. Hangunron, Kyoritu-Shuppan, Tokyo, 1972. (In Japanese)

[9] Tanaka, G. Prefix codes determined by Petri nets. *Algebra Colloquium*, 5(3):255–264, 1998.

[10] Tanaka, G. Circular codes and Petri nets. In *Shoji, K. (ed.) Proc. 9th Symposium on Algebras, Languages and Computation, RIMS Kokyuroku*, 1503:132–138, 2006.

Received 8th May 2008

Acta Cybernetica **19** (2009) 231–244.

On Nilpotent Languages and Their Characterization by Regular Expressions

György Gyurica*

Abstract

Tree languages recognized by deterministic root-to-frontier recognizers are also called DR-languages. The concept of generalized R-chain languages was introduced by the author in his paper *On monotone languages and their characterization by regular expressions* (Acta Cybernetica, **18** (2007), 117-134.) and it has turned out that the monotone DR-languages are exactly those languages that can be given by generalized R-chain languages. In this paper we give a similar characterization for nilpotent DR-languages by means of plain R-chain languages. Also a regular expression based characterization is given for nilpotent string languages.

1 Introduction

Monotone string and DR-languages were characterized by means of regular expressions in [4] and [9]. For string languages it was shown in [4] that the class of monotone languages and the class of languages represented by finite unions of semi-normal chain languages are the same. In case of DR-languages it is turned out in [9] that the class of monotone DR-languages and the class of languages represented by generalized R-chain languages are the same. In this paper our goal was to find a similar characterization for both nilpotent string and DR-languages, however our main focus was directed towards nilpotent DR-languages because nilpotent string languages were already studied in the past intensively.

After introducing the necessary concepts we brought in the concept of plain chain languages by which we characterized nilpotent string languages. Later, a similar chain-like structure was introduced for DR-languages, that were given the name plain R-chain languages. It has turned out that a DR-language is nilpotent if and only if it can be given as a plain R-chain language. The proof required some additional results, among which one states a condition by which the class of DR-languages is closed under x-product. We have also defined when a DR-language is path complete or x-terminating. These concepts turned out to be very handy if we want to characterize the x-product of nilpotent DR-languages.

*Department of Informatics, University of Szeged, Árpád tér 2, H-6720 Szeged, Hungary
E-mail: gyurica@inf.u-szeged.hu

For notions and notation not defined in this paper we refer the reader to [8] and [9].

2 Nilpotent string languages

Let X be a finite nonempty alphabet. X^* denotes the set of all words over X. The *length* of a word $u \in X^*$ is denoted by $|u|$ which is the number of occurrences of letters from X in u. The empty word is denoted by e. As usual, N will denote the set of natural numbers, i.e. $N = \{1, 2, \ldots\}$. $X^+ (= X^* \setminus \{e\})$ denotes the set of words with length greater than 0. The set of words no longer than $k \in N$ is $X^{*,k} = \{u \in X^* : |u| \leq k\}$. A word $w \in X^*$ is a *prefix* of a word $u \in X^*$ if there is a word $v \in X^*$ for which $u = wv$. Moreover, we say that $w \in X^*$ is a *proper* prefix of $u \in X^*$ if w is a prefix of u and $|w| < |u|$.

A system $\mathbf{A} = (A, X, \delta, a_0, A')$ is called an *X-recognizer*, if A is a finite set of *states*, X is the *input alphabet*, $\delta : A \times X \to A$ is the *next-state function*, $a_0 \in A$ is the *initial state*, and $A' \subseteq A$ is the set of *final states*. The next-state function can be extended to a function $\delta^* : A \times X^* \to A$, where $\delta^*(a, e) = a$, and $\delta^*(a, xu) = \delta^*(\delta(a, x), u)$ for every $a \in A$, $x \in X$, $u \in X^*$. If there is no danger of confusion, instead of $\delta^*(a, u)$ we can use the notation $\delta(a, u)$ or simply au.

The *language* $L(\mathbf{A})$ *recognized by* $\mathbf{A}$ is given by

$$L(\mathbf{A}) = \{u \in X^* \mid a_0 u \in A'\}.$$

A language $L \subseteq X^*$ is called *regular* or *recognizable* if it can be recognized by an X-recognizer.

An X-recognizer $\mathbf{A} = (A, X, \delta, a_0, A')$ is *nilpotent* if there is an integer $k \geq 0$ and a state $\bar{a} \in A$ such that $au = \bar{a}$ for all $a \in A$, $u \in X^*$ with $|u| \geq k$. The state $\bar{a}$ is called the *nilpotent element* of $\mathbf{A}$, and the least k for which the above condition holds is called the *degree of nilpotency* of $\mathbf{A}$. A language $L \subseteq X^*$ is *nilpotent* if there is a nilpotent X-recognizer $\mathbf{A}$ for which $L(\mathbf{A}) = L$.

Remark 1. Nilpotent element has various names in the terminology like *absorbent element* (see [2]), or *trap state*, etc. We will use the term *nilpotent element* in the rest of this paper.

The complement of a language $L \subseteq X^*$ is defined as $X^* \setminus L$ and will be denoted by $c(L)$ in the sequel. The following lemma is well-known (see [5]).

Lemma 1. *A language $L \subseteq X^*$ is nilpotent if and only if L or $c(L)$ is finite.*

We introduce some chain-like languages that were used in [4] and [9]. A language $L \subseteq X^*$ is *fundamental*, if $L = Y^*$ for a $Y \subseteq X$. A language $L \subseteq X^*$ is a *chain language*, if L can be given in the form $L = L_0 x_1 L_1 x_2 \ldots x_{k-1} L_{k-1} x_k L_k$, where $x_1, \ldots, x_k \in X$ and every L_i $(0 \leq i \leq k)$ is a product of fundamental languages. We will call a chain language $L = L_0 x_1 L_1 x_2 \ldots x_{k-1} L_{k-1} x_k L_k$ *plain* , if $L_i = \{e\}$ for every $0 \leq i < k$ and $L_k = Y^*$, where $Y = \emptyset$ or $Y = X$. We will use small Greek letters like $\zeta, \eta, \theta, \ldots$ to denote plain chain languages.

Let $\zeta = x_1 x_2 \dots x_k L_k$ be a plain chain language. Clearly, ζ is finite if $L_k = \{e\}$, and ζ is infinite if $L_k = X^*$. Furthermore, the *length* of ζ is defined as $|\zeta| = k$. We consider the plain chain language $\zeta' = x_1 x_2 \dots x_j$ as a *prefix* of ζ if either $j > k$ with $x_{k+1} \dots x_j \in L_k$ or $1 \leq j \leq k$. Note that every word of X^* can be considered as a finite plain chain language.

The following two remarks are trivial.

Remark 2. The languages $\{e\}$ and X^* are nilpotent.

Remark 3. Every finite language can be given as a union of finitely many plain chain languages.

Lemma 2. *Let $L \subseteq X^*$ be an infinite language that is given as a union of plain chain languages $\zeta_1, \dots, \zeta_l$ $(l \in N)$. If for all $u \in X^*$ there is an $i \in \{1, \dots, l\}$ such that u is a prefix of ζ_i, then L is nilpotent.*

Proof. Let us suppose that the conditions of the lemma hold. We construct a recognizer $\mathbf{A} = (X^{*,k} \cup \{\bar{a}\}, X, \delta, \{e\}, A')$ where $k = max\{|\zeta_i| : 1 \leq i \leq l\}$. For every $a \in A$ and $x \in X$, we define the next-state function δ as follows:

$$\delta(a,x) = \begin{cases} ax, & \text{if } a \in X^{*,k-1}, \\ \bar{a}, & \text{if } a \in X^{*,k} \text{ and } |a| = k, \\ \bar{a}, & \text{if } a = \bar{a}. \end{cases}$$

To define the set of final states A', let $\bar{a} \in A'$. Moreover, let $x_1 \dots x_j \in A'$ for every $x_1 \dots x_j L_j \in \{\zeta_1, \dots, \zeta_l\}$, and if $L_j = X^*$ then for every $u \in X^{*,k-j}$ let $x_1 \dots x_j u \in A'$.

Let us now take a word $u \in L$ and let $\zeta \in \{\zeta_1, \dots, \zeta_l\}$ be the shortest plain chain language for which u is a prefix of ζ. If $|u| > k$ then $a_0 u = \bar{a}$ hence $u \in L(\mathbf{A})$. If $|u| \leq k$ then $|\zeta| \leq |u|$ holds since u is represented by ζ in L, therefore by the construction of A' we get $u \in L(\mathbf{A})$. Let us now take a word $w \in L(\mathbf{A})$. The construction of A' implies that there is a plain chain language $\zeta \in \{\zeta_1, \dots, \zeta_l\}$ for which w is a prefix of ζ and $|\zeta| \leq |w|$. Since ζ takes part in the representation of L, we get $w \in L$. Thus $L = L(\mathbf{A})$. It is obvious that $ev = \bar{a}$ for any word $v \in X^*$ with $|v| > k$, therefore L is nilpotent with the nilpotent element $\bar{a}$ and with the degree of nilpotency of $k + 1$. □

Let $\mathbf{A} = (A, X, \delta_{\mathbf{A}}, a_0, A')$ and $\mathbf{B} = (B, X, \delta_{\mathbf{B}}, b_0, B')$ be X-recognizers. The *direct product* of $\mathbf{A}$ and $\mathbf{B}$ is defined as the X-recognizer $\mathbf{A} \times \mathbf{B} = (A \times B, X, \delta, (a_0, b_0), F)$, where $F \subseteq A \times B$ and δ is defined as $\delta((a,b),x) = (\delta_{\mathbf{A}}(a,x), \delta_{\mathbf{B}}(b,x))$ for all $a \in A, b \in B$ and $x \in X$. Let τ be a mapping from A onto B. We say that τ is *homomorphism* of $\mathbf{A}$ onto $\mathbf{B}$ if $\tau(a_0) = b_0$, $\tau^{-1}(B') = A'$ and $\tau(\delta_{\mathbf{A}}(a,x)) = \delta_{\mathbf{B}}(\tau(a),x)$ for every $a \in A$ and $x \in X$. In this case $\mathbf{B}$ is the *homomorphic image* of $\mathbf{A}$ beside τ.

The following properties of nilpotent recognizers are well-known.

Lemma 3. *The direct products and homomorphic images of nilpotent recognizers are also nilpotent.*

Proof. It is obvious that the class of nilpotent recognizers is closed under direct products (see [5]).

To prove that the class of nilpotent recognizers is closed under the homomorphic images, let the recognizer $\mathbf{B} = (B, X, \delta_{\mathbf{B}}, b_0, B')$ be the homomorphic image of the nilpotent recognizer $\mathbf{A} = (A, X, \delta_{\mathbf{A}}, a_0, A')$ under a homomorphism $\tau : A \to B$. Let the state $\bar{a} \in A$ be the nilpotent element of $\mathbf{A}$ and let k be the degree of nilpotency of $\mathbf{A}$. Let $\tau(\bar{a}) = \bar{b}$. Taking any word $u \in X^*$ for which $|u| \geq k$ and an element $a \in A$ we get that $au = \bar{a}$, and thus $\tau(a)u = \tau(\bar{a})$. As every element of B has an inverse image in A we can state that $\mathbf{B}$ is nilpotent with the degree of nilpotency of k and with the nilpotent element $\bar{b}$. □

It is a widely used fact that the unions and the intersections of languages recognized by given recognizers can be recognized by direct products of these recognizers (see [5]). Also it is clear that the complement of a nilpotent language is also nilpotent (again, see [5]). Thus we get

Corollary 1. *The family of nilpotent languages is closed under union, intersection and complement.*

Let $\mathbf{A} = (A, X, \delta_{\mathbf{A}}, a_0, A')$ be an X-recognizer. The X-recognizer $\mathbf{B} = (B, X, \delta_{\mathbf{B}}, b_0, B')$ is the *connected subrecognizer* of $\mathbf{A}$, if $B = \{a_0 u | u \in X^*\}$, $B' = A' \cap B$, $a_0 = b_0$ and $\delta_{\mathbf{B}}(b, x) = \delta_{\mathbf{A}}(b, x)$ for every $b \in B, x \in X$.

Lemma 4. *The connected subrecognizer of a nilpotent recognizer is nilpotent.*

Proof. Let $\mathbf{A} = (A, X, \delta_{\mathbf{A}}, a_0, A')$ be a nilpotent X-recognizer with the nilpotent element of $\bar{a}$ and with the degree of nilpotency of k. Also let $\mathbf{B} = (B, X, \delta_{\mathbf{B}}, b_0, B')$ be the connected subrecognizer of $\mathbf{A}$. By the definitions of connected subrecognizer and nilpotent X-recognizer we get that $\bar{a} \in B$. Again, by the definition of connected subrecognizer we obtain that $b_0 u = \bar{a}$ for every word $u \in X^*$ with $|u| \geq k$. Thus $\mathbf{B}$ is nilpotent. □

It is also a well-known fact that the minimal recognizer recognizing a language L is a homomorphic image of the connected subrecognizer of any recognizer recognizing L (see [8]). Thus we have

Corollary 2. *A language is nilpotent iff the minimal recognizer recognizing it is nilpotent.*

Before we continue studying nilpotent languages, we observe some basic correlations between nilpotent and monotone languages.

An X-recognizer $\mathbf{A} = (A, X, \delta, a_0, A')$ is *monotone* if there is a partial ordering $\leq$ on A such that $a \leq \delta(a, x)$ holds for all $a \in A$ and $x \in X$. It is obvious that for all $a \in A$ and $u \in X^*$, $a \leq au$ holds, too. A language $L \subseteq X^*$ is *monotone* if $L = L(\mathbf{A})$ for a monotone X-recognizer $\mathbf{A}$. Later we will use the fact that every partial ordering on a finite set can be extended to a linear ordering. For more details on monotone languages we refer the reader to [4].

The following property of nilpotent languages is well-known (see [10]).

Lemma 5. *Every nilpotent language is monotone.*

Proof. Let $L \subseteq X^*$ be a nilpotent language and let $\mathbf{A} = (A, X, \delta, a_0, A')$ be a nilpotent recognizer for which $L = L(\mathbf{A})$. Moreover, let us suppose that L is not monotone. This means that there are states $a, b \in A$ with $a \neq b$ and words $u, v \in X^*$ such that $au = b$ and $bv = a$. Let k be the degree of nilpotency of $\mathbf{A}$. Then $a(uv)^k = a$ and $b(vu)^k = b$, which contradicts the assumption that $\mathbf{A}$ is nilpotent with degree of nilpotency of k. □

Remark 4. In the proof of Lemma 5 we relied on the assumption that there are states a and b and words u and v such that $a \neq b$, $au = b$ and $bv = a$. If there is no such a pair of states a and b, then the relation defined by $a' \leq a'u$ for every $a' \in A$ and $u \in X^*$, is a monotone order.

From Lemma 5 we easily get

Corollary 3. *Let $\mathbf{A} = (A, X, \delta, a_0, A')$ be a nilpotent recognizer. There is a linear ordering $\leq$ on A such that $a \leq ax$ holds for all $a \in A$ and $x \in X$.*

The converse of Lemma 5 does not hold as the following example shows.

Example 1. It is easy to see that $L = ab^*$ is monotone, but is not nilpotent.

In the sequel, we will characterize nilpotent languages by means of plain chain languages.

Lemma 6. *Every nilpotent language $L \subseteq X^*$ can be given as a union of finitely many plain chain languages $\zeta_1, \ldots, \zeta_l$ ($l \in N$), where in case of infinite L it holds that for all $u \in X^*$ there is an $i \in \{1, \ldots, l\}$ such that u is a prefix of ζ_i.*

Proof. Let $L \subseteq X^*$ be a nilpotent language. If L is finite, then by Remark 3 L can be given as a union of finitely many plain chain languages. If L is infinite, then let $\mathbf{A} = (A, X, \delta, a_0, A')$ be the minimal nilpotent recognizer for which $L = L(\mathbf{A})$. If A is singleton, then $L(\mathbf{A}) = X^*$, thus L is a plain chain language. Let us now assume that $A = \{a_0, \ldots, a_n\}$ ($n > 0$), and let the state a_n be the nilpotent element of $\mathbf{A}$. Using Corollary 3, we have a linear ordering $\leq$ on A such that $a_0 \leq \ldots \leq a_n$ holds. Since $\mathbf{A}$ is nilpotent and minimal, $a \neq a_n$ implies $ax \neq a$ for all $a \in A$ and $x \in X$. Let us define the recognizer $\mathbf{A}_{i,j} = (A, X, \delta, a_i, \{a_j\})$ for all $0 \leq i, j \leq n$. Furthermore, for all $x \in X$, let us define the recognizer $\mathbf{A}_x = (A, X, \delta, a_0, A_x)$, where $A_x = \{a \in A \setminus \{a_n\} \mid ax = a_n\}$. Using the recognizers defined above we can write $L(\mathbf{A}_{0,n}) = \bigcup_{x \in X} L(\mathbf{A}_x) x L(\mathbf{A}_{n,n})$. Since $L(\mathbf{A}_x)$ is finite and $L(\mathbf{A}_{n,n}) = X^*$, we get that $L(\mathbf{A}_{0,n})$ is given as a finite union of plain chain languages. Using the languages $L(\mathbf{A}_{i,j})$, we can give L as

$$L = L(\mathbf{A}) = \bigcup_{a_m \in A'} L(\mathbf{A}_{0,m}) = \bigcup_{a_m \in A' \setminus \{a_n\}} L(\mathbf{A}_{0,m}) \cup \bigcup_{x \in X} L(\mathbf{A}_x) x L(\mathbf{A}_{n,n}),$$

where every $L(\mathbf{A}_{0,m})$ is finite ($0 \leq m < n$), hence by Remark 3 we gave L as the union of finitely many plain chain languages. Let us now take a word $u \in X^*$ and

let $w \in X^*$ be any word for which $|uw| \geq n$. Since $a_0uw = a_n$ there is a plain chain language ζ in the above representation of $L(\mathbf{A}_{0,n})$ for which u is the prefix of ζ. □

From Lemma 1, Lemma 2 and Lemma 6 we directly obtain

Theorem 1. *Let $L \subseteq X^*$ be a regular language. L is nilpotent iff L can be given as a union of finitely many plain chain languages $\zeta_1, \ldots, \zeta_l$ $(l \in N)$, where in case of infinite L it holds that for all $u \in X^*$ there is an $i \in \{1, \ldots, l\}$ such that u is a prefix of ζ_i.*

3 Nilpotent DR-languages

A finite nonempty set of operational symbols is called *ranked alphabet* and will be denoted by Σ in this paper. The subset of all m-ary operational symbols will be denoted by Σ_m $(\subseteq \Sigma)$, $m > 0$. We shall exclude the case $m = 0$, so it is supposed that $\Sigma_0 = \emptyset$ in the sequel.

Let X be a set of variables. The set $T_\Sigma(X)$ of ΣX*-trees* is defined as follows:

(i) $X \subseteq T_\Sigma(X)$,

(ii) $\sigma(p_1, \ldots, p_m) \in T_\Sigma(X)$ if $m \geq 0$, $\sigma \in \Sigma_m$ and $p_1, \ldots, p_m \in T_\Sigma(X)$,

(iii) every ΣX-tree can be obtained by applying the rules (i) and (ii) a finite number of times.

In the rest of this paper X will stand for the countable set $\{x_1, x_2, \ldots\}$, and for every $n \geq 0$, X_n will denote the subset $\{x_1, \ldots, x_n\} \subseteq X$. The power set of the set S will be denoted by $\mathfrak{p}(S)$.

A *deterministic root-to-frontier Σ-algebra* (or *DR Σ-algebra* for short) is a pair $\mathcal{A} = (A, \Sigma)$, where A is a nonempty set and Σ is a ranked alphabet. Every $\sigma \in \Sigma_m$ is represented as a mapping $\sigma^{\mathcal{A}} : A \to A^m$. We call $\mathcal{A}$ *finite*, if A is finite.

A system $\mathfrak{A} = (\mathcal{A}, a_0, \mathbf{a})$ will represent a *deterministic root-to-frontier ΣX_n-recognizer* (or a *DR ΣX_n-recognizer* for short), where $\mathcal{A} = (A, \Sigma)$ is a finite DR Σ-algebra, $a_0 \in A$ is the *initial state*, and $\mathbf{a} = (A^{(1)}, \ldots, A^{(n)}) \in \mathfrak{p}(A)^n$ is the *final state vector*. If Σ or X_n is not specified, we speak of *DR-recognizers*.

Let $\mathfrak{A} = (\mathcal{A}, a_0, \mathbf{a})$ be a DR ΣX_n-recognizer. Let us define the mapping $\alpha : T_\Sigma(X_n) \to \mathfrak{p}(A)$ as usual. For every $p \in T_\Sigma(X_n)$

(i) if $p = x_i \in X_n$, then $\alpha(p) = A^{(i)}$,

(ii) if $p = \sigma(p_1, \ldots, p_m)$, then $\alpha(p) = \{a \in A \mid \sigma^{\mathcal{A}}(a) \in \alpha(p_1) \times \ldots \times \alpha(p_m)\}$.

The *tree language recognized* by $\mathfrak{A}$ is denoted by $T(\mathfrak{A})$ and is given by

$$T(\mathfrak{A}) = \{p \in T_\Sigma(X_n) \mid a_0 \in \alpha(p)\}.$$

Tree languages recognized by DR-recognizers are also called *DR-languages*.

Let $\mathfrak{A}$ be a DR ΣX_n-recognizer and $a \in A$ one of its states. The tree language recognized by $\mathfrak{A}$ from the state a is defined by

$$T(\mathfrak{A}, a) = \{\, p \in T_\Sigma(X_n) \mid a \in \alpha(p)\}.$$

A state a is called 0-*state* if $T(\mathfrak{A}, a) = \emptyset$. $\mathfrak{A}$ is called *normalized* if for all $\sigma \in \Sigma_m$ and $a \in A$ it holds that each component of $\sigma^{\mathcal{A}}(a)$ is a 0-state or no component of $\sigma^{\mathcal{A}}(a)$ is a 0-state. Moreover, $\mathfrak{A}$ is called *reduced* if for all states $a, b \in A$ it holds that $a \neq b$ implies $T(\mathfrak{A}, a) \neq T(\mathfrak{A}, b)$. It is a well-known fact that every DR-language can be recognized by a normalized and reduced DR-recognizer. For more details we refer the reader to [6], [7] and [8].

Now we define the ordinary alphabet $\hat{\Sigma}$ corresponding to the ranked alphabet Σ. For all $\sigma, \tau \in \Sigma$, let

(i) $\hat{\Sigma}_\sigma = \{\sigma_1, \ldots, \sigma_m\}$, if $\sigma \in \Sigma_m$ $(m > 0)$, and

(ii) $\hat{\Sigma}_\sigma \cap \hat{\Sigma}_\tau = \emptyset$, if $\sigma \neq \tau$.

We define $\hat{\Sigma}$ as $\hat{\Sigma} = \bigcup (\hat{\Sigma}_\sigma \mid \sigma \in \Sigma)$.

For each $x \in X_n$, the set $g_x(t)$ of x-*paths* of a tree $t \in T_\Sigma(X_n)$ is defined as follows:

(i) $g_x(x) = \{e\}$,

(ii) $g_x(y) = \emptyset$ for $y \in X_n, x \neq y$,

(iii) $g_x(t) = \sigma_1 g_x(t_1) \cup \ldots \cup \sigma_m g_x(t_m)$ for $t = \sigma(t_1, \ldots, t_m)$, $\sigma \in \Sigma_m$, $t_i \in T_\Sigma(X_n)$, $1 \leq i \leq m$, $m > 0$.

The mappings g_x are extended to ΣX_n-tree languages in the natural way, that is, for any tree language $T \subseteq T_\Sigma(X_n)$ and variable $x \in X_n$, let $g_x(T) = \bigcup_{t \in T} g_x(t)$. The sets $g_x(T) \subseteq \hat{\Sigma}^*$ are also denoted by T_x and are called the *path languages* of T. Moreover, let us define $g(T)$ as $g(T) = \bigcup_{x \in X} T_x$. A tree language $T \subseteq T_\Sigma(X_n)$ is said to be *closed* if a tree $t \in T_\Sigma(X_n)$ is in T if and only if $g_x(t) \subseteq T_x$ for all $x \in X_n$. It is a well-known result that a regular tree language is DR-recognizable if and only if it is closed (cf. [1] and [11]).

For any integer $n \in N$ and sets $S_1, \ldots, S_n$, let $\pi_i : S_1 \times \ldots \times S_n \to S_i$ be the i-th projection, that is, $\pi_i(s_1, \ldots, s_i, \ldots, s_n) = s_i$ for all $s_i \in S_i$ and $1 \leq i \leq n$. Let Σ be a ranked alphabet, and let $\hat{\Sigma}$ be the alphabet corresponding to it. Let $\mathcal{A} = (A, \Sigma)$ be a DR Σ-algebra. For every $u \in \hat{\Sigma}^*$, the mapping $u^{\mathcal{A}} : A \to A$ is defined as follows:

(i) If $u = e$, then $au^{\mathcal{A}} = a$, and

(ii) if $u = \sigma_j v$, then $au^{\mathcal{A}} = \pi_j(\sigma(a))v^{\mathcal{A}}$ for all $a \in A$, $\sigma \in \Sigma_m$, $v \in \hat{\Sigma}^*$, and $j \in \{1, \ldots, m\}$.

The mapping defined above can be extended to subsets of $\hat{\Sigma}^*$ in the natural way. In the rest of this paper we will omit the superscript $\mathcal{A}$ in $u^{\mathcal{A}}$ if the DR Σ-algebra $\mathcal{A}$ inducing $u^{\mathcal{A}}$ is obvious.

A DR Σ-algebra $\mathcal{A} = (A, \Sigma)$ is *nilpotent*, if there are an integer $k \geq 0$ and an element $\bar{a} \in A$ such that $au = \bar{a}$ for all $a \in A$ and $u \in \hat{\Sigma}^*$ with $|u| \geq k$. The state $\bar{a}$ is called the *nilpotent element* of $\mathcal{A}$ and the least k for which the above condition holds is called the *degree of nilpotency* of $\mathcal{A}$. A DR ΣX_n-recognizer $\mathfrak{A} = (\mathcal{A}, a_0, \mathbf{a})$ is *nilpotent* if the underlying DR Σ-algebra $\mathcal{A}$ is nilpotent. Finally, a ΣX_n-tree language T is *nilpotent* if it can be recognized by a nilpotent DR ΣX_n-recognizer.

Remark 5. A different definition of nilpotent DR Σ-algebras and a typical characterization can be found in [3]. We will show that the two definitions define the same set of DR Σ-algebras.

Let $\mathcal{A} = (A, \Sigma)$ be a DR Σ-algebra, let $a \in A$ be an element and let $t \in T_\Sigma(X_n)$ be a tree. We define the word $\overline{\mathrm{fr}}(at) \in A^*$ as follows:

(i) if $t \in X$, then $\overline{\mathrm{fr}}(at) = a$,

(ii) if $t = \sigma(t_1, \ldots, t_m)$, then $\overline{\mathrm{fr}}(at) = \overline{\mathrm{fr}}(a_1 t_1) \ldots \overline{\mathrm{fr}}(a_m t_m)$, where $\sigma \in \Sigma_m$, $\sigma^{\mathcal{A}}(a) = (a_1, \ldots, a_m)$, $t_1, \ldots, t_m \in T_\Sigma(X_n)$, $m > 0$.

For any tree $t \in T_\Sigma(X_n)$, let $\mathrm{mh}(t) = min\{|u| : u \in g(t)\}$, that is, $\mathrm{mh}(t)$ is the length of the shortest path leading from the root of t to a leaf. Now we recall the definition of nilpotent DR Σ-algebra from [3]. A DR Σ-algebra $\mathcal{A} = (A, \Sigma)$ is *nilpotent* if there are an integer $k \geq 0$ and an element $\bar{a} \in A$ such that for all $a \in A$ and $t \in T_\Sigma(X_n)$ with $\mathrm{mh}(t) \geq k$, $\overline{\mathrm{fr}}(at) = \bar{a}^l$ for a natural number l. This $\bar{a}$ is called the *nilpotent element* of $\mathcal{A}$ and the least k for which the above condition holds is called the *degree of nilpotency* of $\mathcal{A}$.

Lemma 7. *The two definitions of nilpotent DR languages above define the same set of DR Σ-algebras.*

Proof. Let $\mathcal{A} = (A, \Sigma)$ be a DR Σ-algebra, let k be an integer, and let $\bar{a} \in A$ be an element such that for any $a \in A$ and $t \in T_\Sigma(X_n)$ with $\mathrm{mh}(t) \geq k$ we have $\overline{\mathrm{fr}}(at) = \bar{a}^l$ for a natural number l. Let us now take a word $u \in \hat{\Sigma}^*$ such that $|u| \geq k$. By taking any tree $p \in T_\Sigma(X_n)$ for which u is the shortest path in $g(p)$ we have $\overline{\mathrm{fr}}(ap) = \bar{a}^{l'}$ for a natural number l'. That means $au = \bar{a}$.

Conversely, let $k \geq 0$ be an integer and $\bar{a} \in A$ a state such that for every $a \in A$ and $u \in \hat{\Sigma}^*$ with $|u| \geq k$, $au = \bar{a}$ holds. Let us now take a tree $p \in T_\Sigma(X_n)$ for which $\mathrm{mh}(p) \geq k$. Since every path in $g(p)$ is at least k long, we have $\overline{\mathrm{fr}}(ap) = \bar{a}^l$ for a natural number l. Therefore the two definitions define the same set of DR Σ-algebras. □

A DR Σ-algebra $\mathcal{A} = (A, \Sigma)$ is called *monotone* if there is a partial ordering $\leq$ on A such that $a \leq \pi_i(\sigma(a))$ for all $a \in A$, $\sigma \in \Sigma_m$ and $1 \leq i \leq m$. Moreover, we say that a DR ΣX_n-recognizer $\mathfrak{A}$ is a *monotone DR ΣX_n-recognizer* if the underlying

DR Σ-algebra $\mathcal{A}$ is monotone. Finally, a language $T \subseteq T_\Sigma(X_n)$ is *monotone*, if $T = T(\mathfrak{A})$ for a monotone DR ΣX_n-recognizer $\mathfrak{A}$.

As in the string case, there is a basic correlation between the nilpotent and monotone DR-languages that we state in

Lemma 8. *Every nilpotent DR-language is monotone.*

Corollary 4. *Let $\mathfrak{A} = (\mathcal{A}, a_0, \mathbf{a})$ be a nilpotent DR ΣX_nrecognizer where $\mathcal{A} = (A, \Sigma)$. There exists a linear ordering $\leq$ on A such that $a \leq \pi_i(\sigma(a))$ holds for all $a \in A$, $\sigma \in \Sigma_m$ and $i \in \{1, \ldots, m\}$.*

Similarly to the string case, the converse of Lemma 8 does not hold.

4 Simple approach

Before we continue, we have to clarify some details regarding some particular operations on tree languages. The *σ-product* of ΣX_n-tree languages $T_1, \ldots, T_m$ is the tree language $\sigma(T_1, \ldots, T_m) = \{\sigma(t_1, \ldots, t_m) \mid t_i \in T_i,\ 1 \leq i \leq m\}$, where $m > 0$ and $\sigma \in \Sigma_m$. We assume that the reader is familiar with the operations of union, x-product and x-iteration. In the sequel, we use the operation of x-product in right-to-left manner, that is, for any tree languages $S, T \subseteq T_\Sigma(X_n)$ the x-product $T \cdot_x S$ is interpreted as a tree language in which the trees are obtained by taking a tree s from S and replacing every leaf symbol x in s by a tree from T. Note that different occurrences of x may be replaced by different trees from T. We will also assume that $T \cdot_y R \cdot_x S$ always means $T \cdot_y (R \cdot_x S)$ for any variables $x, y \in X_n$ and tree languages $S, R, T \subseteq T_\Sigma(X_n)$.

Let Σ be a ranked alphabet and let X_n be a set of variables. The set $RE(\Sigma X_n)$ of all *regular ΣX_n-expressions* and the tree language $T(\eta)$ *represented* by $\eta \in RE(\Sigma X_n)$ are defined in parallel as follows:

- $\emptyset \in RE(\Sigma X_n)$, $\quad T(\emptyset) = \emptyset$,
- $\forall x \in X_n : x \in RE(\Sigma X_n)$, $\quad T(x) = \{x\}$,

If $\sigma \in \Sigma_m$, $\eta_1, \eta_2, \ldots, \eta_m \in RE(\Sigma X_n)$, $x \in X_n$, $m > 0$, then

- $(\eta_1) + (\eta_2) \in RE(\Sigma X_n)$, $\quad T((\eta_1) + (\eta_2)) = T(\eta_1) \cup T(\eta_2)$,
- $(\eta_2) \cdot_x (\eta_1) \in RE(\Sigma X_n)$, $\quad T((\eta_2) \cdot_x (\eta_1)) = T(\eta_2) \cdot_x T(\eta_1)$,
- $(\eta_1)^{*,x} \in RE(\Sigma X_n)$, $\quad T((\eta_1)^{*,x}) = T(\eta_1)^{*,x}$,
- $\sigma(\eta_1, \ldots, \eta_m) \in RE(\Sigma X_n)$, $\quad T(\sigma(\eta_1, \ldots, \eta_m)) = \sigma(T(\eta_1), \ldots, T(\eta_m))$.

Some parentheses can be omitted from regular ΣX_n-expressions, if a precedence relation is assumed between the operations of σ-product, x-iteration, x-product, and union in the given order.

Let $\mathfrak{A} = (\mathcal{A}, a_0, \mathbf{a})$ be a nilpotent DR ΣX_n-recognizer, where $\mathcal{A} = (A, \Sigma)$, $A = \{a_0, \ldots, a_k\}$ and $\mathbf{a} = (A^{(1)}, \ldots, A^{(n)})$. Due to Corollary 4 we assume that $a_0 \leq$

$a_1 \leq \ldots \leq a_k$ holds and we can also suppose that a_k is the nilpotent element of $\mathfrak{A}$. Let $\Xi_k = \{\xi_0, \ldots, \xi_k\}$ be a set of auxiliary variables for which $X_n \cap \Xi_k = \emptyset$ holds, and let $\phi : A \to \Xi_k$ be a bijective mapping defined by $\phi(a_i) = \xi_i$ for every $0 \leq i \leq k$. Since every nilpotent DR-language is monotone, we may recall the *trivial regular expression belonging to* $\mathfrak{A}$ ($\eta_{\mathfrak{A}}$ for short), which is defined in [9] as follows:

$$\eta_{\mathfrak{A}} = \eta_k \cdot_{\xi_k} \eta_{k-1} \cdot_{\xi_{k-1}} \cdots \cdot_{\xi_1} \eta_0,$$

where for each $i = 0, \ldots, k$,

$$\eta_i = (p_1^i + \cdots + p_{l_i}^i + y_1^i + \cdots + y_{r_i}^i) \cdot_{\xi_i} (t_1^i + \cdots + t_{j_i}^i)^{*,\xi_i},$$

and where

1) $y_1^i, \ldots, y_{r_i}^i$ are all the elements of the set $\{x_z \in X_n |\ a_i \in A^{(z)}, 1 \leq z \leq n\}$,

2) $p_s^i = \sigma(\xi_{i_1}, \ldots, \xi_{i_m})$ for such $\sigma \in \Sigma_m$ and $\xi_{i_v} \in \Xi_k$ $(1 \leq v \leq m)$ that $\sigma(a_i) = (\phi^{-1}(\xi_{i_1}), \ldots, \phi^{-1}(\xi_{i_m}))$ and $a_i \notin \bigcup_{1 \leq v \leq m} \{\pi_v(\sigma(a_i))\}$ hold for every $s \in \{1, \ldots, l_i\}$,

3) $t_s^i = \sigma(\xi_{i_1}, \ldots, \xi_{i_m})$ for such $\sigma \in \Sigma_m$ and $\xi_{i_v} \in \Xi_k$ $(1 \leq v \leq m)$ that $\sigma(a_i) = (\phi^{-1}(\xi_{i_1}), \ldots, \phi^{-1}(\xi_{i_m}))$ and $a_i \in \bigcup_{1 \leq v \leq m} \{\pi_v(\sigma(a_i))\}$ hold for every $s \in \{1, \ldots, j_i\}$,

4) $|\{p_1^i, \ldots, p_{l_i}^i\}| + |\{t_1^i, \ldots, t_{j_i}^i\}| = |\Sigma|$.

In each η_i the part $(p_1^i + \cdots + p_{l_i}^i + y_1^i + \cdots + y_{r_i}^i)$ is called the *terminating part* of η_i, furthermore, the part $(t_1^i + \cdots + t_{j_i}^i)^{*,\xi_i}$ is called the *iterating part* of η_i. The expressions of the form $\eta_k \cdot_{\xi_k} \ldots \eta_1 \cdot_{\xi_1} \eta_0$ are called *chains*.

Let us now observe the regular $\Sigma(X_n \cup \Xi_k)$-expression $\eta_{\mathfrak{A}}$ that is detailed above. It is obvious that in each η_i $(0 \leq i < k)$ the iterating part is empty because there is no symbol $\sigma \in \Sigma_m$ and state $a \in A \setminus \{a_k\}$ for which $a \in \bigcup_{1 \leq v \leq m} \{\pi_v(\sigma(a))\}$. Thus we can omit these iterating parts from $\eta_{\mathfrak{A}}$. By these omissions we simplified the trivial regular expression belonging to $\mathfrak{A}$, and we will call the result the *plain regular expression belonging to* $\mathfrak{A}$ (denoted by $\zeta_{\mathfrak{A}}$).

5 Characterization

Let $S \subseteq T_\Sigma(X_n)$ be a tree language and let $p \in T_\Sigma(X_n)$ be a tree. The *height* $height(p)$, *root* $root(p)$, *leaves* $leaves(p)$ and the set of *subtrees* $Sub(p)$ of the tree p are defined as follows:

(i) If $p \in X_n$, then $height(p) = 0$, $root(p) = p$, $leaves(p) = \{p\}$, and $Sub(p) = \{p\}$.

(ii) If $p = \sigma(t_1, \ldots, t_m)$, $\sigma \in \Sigma_m$, $t_i \in T_\Sigma(X_n)$, $1 \leq i \leq m$, $m > 0$, then $height(p) = 1 + max\{height(t_i) : 1 \leq i \leq m\}$, $root(p) = \sigma$, $leaves(p) = \bigcup_{1 \leq i \leq m} leaves(t_i)$, and $Sub(p) = \{p\} \cup \bigcup_{1 \leq i \leq m} (Sub(t_i))$.

The above functions (except height) are extended from trees to tree languages as follows: $root(S) = \{root(p) \mid p \in S\}$, $leaves(S) = \bigcup_{p \in S} leaves(p)$, and $Sub(S) = \bigcup_{p \in S} Sub(p)$.

For any tree language $S \subseteq T_\Sigma(X_n)$, let the set of *operational symbols appearing in* S be defined as $root(Sub(S)) \setminus X_n$ and let it be denoted by Σ_S. For any language $S \subseteq T_\Sigma(X_n)$ and variable $x \in X_n$, let $\Sigma_{S,x}$ denote the set $\{\sigma \in \Sigma_m \mid \exists u \in g_x(S),\ \exists v \in \hat{\Sigma}^*,\ \exists i \in \{1, \ldots, m\} : u\sigma_i v \in g(S),\ m > 0\}$.

Later we will use the following lemma.

Lemma 9. *Every finite DR-language is nilpotent*

Proof. Let T be a finite DR-language and let us assume that the DR ΣX_n-recognizer $\mathfrak{A}$ recognizes T. It is obvious that $\mathfrak{A}$ is nilpotent with the degree of nilpotency of $1 + max\{height(t) : t \in T\}$. □

The following lemma is similar to Theorem 18 in [9] with the only difference that monotonicity is not included.

Lemma 10. *Let S and T be DR-languages, and let $x \in X_n$. If $root(T) \cap \Sigma_{S,x} = \emptyset$, then $T \cdot_x S$ is deterministic.*

Proof. The proof is the same as the proof of Theorem 18 in [9] except that we have to omit monotonicity from the conditions and conclusion. □

We say that a tree language $S \subseteq T_\Sigma(X_n)$ is *path complete* if for every word $u \in g(S)$ and for every prefix $w = w_1 \ldots w_{l-1} w_l$ of u, the word $w_1 \ldots w_{l-1} \bar{w}$ is a prefix of a word from $g(S)$ for all $\bar{w}, w_1, \ldots, w_l \in \hat{\Sigma}$, $l \in N$.

Lemma 11. *Let $x \in X_n$ be a variable and let $S \subseteq T_\Sigma(X_n)$, $T \subseteq T_\Sigma(X_n)$, and $T \cdot_x S$ be DR-languages. If S and T are path complete, then so is $T \cdot_x S$.*

Proof. Let the conditions of the lemma hold. Let us take a word u from $g(T \cdot_x S)$ and take a prefix $w = w_1 \ldots w_{l-1} w_l$ of u where $w_1, \ldots, w_l \in \hat{\Sigma}$ and $l \in N$. Let $\bar{w} \in \hat{\Sigma}$ be also arbitrarily chosen. If $u \in g(S)$ then $w_1 \ldots w_{l-1} \bar{w}$ is a prefix of a word from $g(S)$ because S is path complete, and so $w_1 \ldots w_{l-1} \bar{w}$ is a prefix of a word from $g(T \cdot_x S)$. If $u = u_S u_T$ where $u_S \in g_x(S)$ and $u_T \in g(T)$, then we differentiate 3 cases:

(i) If w is a prefix of u_S then $w_1 \ldots w_{l-1} \bar{w}$ is a prefix of a word from $g(S)$ because S is path complete. Thus $w_1 \ldots w_{l-1} \bar{w}$ is a prefix of a word from $g(T \cdot_x S)$.

(ii) If $u_S = e$ then $w_1 \ldots w_{l-1} \bar{w}$ is a prefix of a word from $g(T)$ since T is path complete. Hence $w_1 \ldots w_{l-1} \bar{w}$ is a prefix of a word from $g(T \cdot_x S)$.

(iii) If u_S is a prefix of w then there is an integer $i \in N$ such that $u_S = w_1 \ldots w_i$. In this case $w_{i+1} \ldots w_{l-1} \bar{w}$ is a prefix of a word from $g(T)$. Since $u_S \in g_x(S)$ $w_1 \ldots w_i w_{i+1} \ldots w_{l-1} \bar{w}$ is a prefix of a word from $g(T \cdot_x S)$.

Since in every case $w_1 \ldots w_{l-1} \bar{w}$ is a prefix of a word from $g(T \cdot_x S)$, we proved that $T \cdot_x S$ is path complete. □

For any variable $x \in X_n$, a tree language $S \subseteq T_\Sigma(X_n)$ is said to be *x-terminating* if the following condition holds. For every $u \in g(S)$, if u is not a proper prefix of any $w \in g(S)$, then $u \in g_x(S)$.

Theorem 2. *Let $x_i \in X_n$ be a variable and let $S \subseteq T_\Sigma(X_n)$ and $T \subseteq T_\Sigma(X_n)$ be nilpotent DR-languages. If $root(T) \cap \Sigma_{S,x_i} = \emptyset$ and S is finite, path complete and x_i-terminating, then $T \cdot_{x_i} S$ is nilpotent.*

Proof. Let the conditions of the theorem hold. Let $\mathfrak{A} = (\mathcal{A}, a_0, \mathbf{a})$ and $\mathfrak{B} = (\mathcal{B}, b_0, \mathbf{b})$ be reduced, connected and normalized nilpotent DR ΣX_n-recognizers, where $\mathcal{A} = (A, \Sigma)$, $\mathbf{a} = (A^{(1)}, \ldots, A^{(n)})$, $\mathcal{B} = (B, \Sigma)$, $\mathbf{b} = (B^{(1)}, \ldots, B^{(n)})$ and $A \cap B = \emptyset$ such that $T(\mathfrak{A}) = S$ and $T(\mathfrak{B}) = T$. Let k and l be the degrees of nilpotency of $\mathfrak{A}$ and $\mathfrak{B}$, respectively, and also let $\bar{a}$ and $\bar{b}$ be the nilpotent elements of $\mathfrak{A}$ and $\mathfrak{B}$, respectively.

We construct a nilpotent DR ΣX_n-recognizer $\mathfrak{C} = (\mathcal{C}, c_0, \mathbf{c})$ that recognizes $T \cdot_{x_i} S$. Let $\mathcal{C} = (C, \Sigma)$, $C = (A \cup B) \setminus \{\bar{a}\}$, $c_0 = a_0$ and $\mathbf{c} = (C^{(1)}, \ldots, C^{(n)})$, where $\mathbf{c}$ is defined in the following way:

$$C^{(j)} = \begin{cases} A^{(j)} \cup B^{(j)} \cup A^{(i)}, & \text{if } x_j \in T,\ j \neq i, \\ A^{(j)} \cup B^{(j)}, & \text{if } x_j \notin T,\ j \neq i, \\ B^{(j)} \cup A^{(i)}, & \text{if } x_j \in T,\ j = i, \\ B^{(j)}, & \text{if } x_j \notin T,\ j = i. \end{cases}$$

The elements of Σ in $\mathcal{C}$ are represented as follows. For all $\sigma \in \Sigma$ and $c \in C$, let

$$\sigma^{\mathcal{C}}(c) = \begin{cases} \sigma^{\mathcal{B}}(c), & \text{if } c \in B, \\ \sigma^{\mathcal{B}}(b_0), & \text{if } c \in A^{(i)},\ \sigma \in root(T), \\ \sigma^{\mathcal{B}}(b_0), & \text{if } c \in A^{(i)},\ cu^{\mathcal{A}} = \bar{a} \text{ for any } u \in \hat{\Sigma}, \\ \sigma^{\mathcal{A}}(c), & \text{otherwise.} \end{cases}$$

First we show that $T(\mathfrak{C}) = T \cdot_{x_i} S$. In order to show $T(\mathfrak{C}) \subseteq T \cdot_{x_i} S$, let us consider the definition of $\mathbf{c}$. Obviously, we need to keep $B^{(j)}$ in $C^{(j)}$ in all cases. Then, in case of $j \neq i$ we need to keep $A^{(j)}$ in $C^{(j)}$ to retain all x_j-paths in $T(\mathfrak{C})$ that we had in $T(\mathfrak{A})$. Finally, if $x_j \in T$, then we need to derive x_j in all states of $A^{(i)}$ in $\mathfrak{C}$ since in this case every path from $g_{x_i}(S)$ is in $g_{x_j}(T \cdot_{x_i} S)$ as well. To show $T \cdot_{x_i} S \subseteq T(\mathfrak{C})$, let us consider the definition of $\sigma^{\mathcal{C}}(c)$ which is consisted of four parts. In the first we guarantee a $\mathfrak{B}$-like processing in $\mathfrak{C}$. The second and the third cases ensure that the processing of a word from $T \cdot_{x_i} S$, that is in a state $a \in A^{(i)}$ at the moment, can be continued in $\mathfrak{B}$. This is important because for any variable $x_j \in X_n$ and every path $uv \in g_{x_j}(T \cdot_{x_i} S)$ with $v \in g_{x_j}(T)$ and $u \in g_{x_i}(S)$ we need $c_0 uv \in C^{(j)}$. Finally, the fourth case manages the processing of any path $g(S)$ in $\mathfrak{C}$. Also, the condition $root(T) \cap \Sigma_{S,x_i} = \emptyset$ guarantees us that $\mathfrak{C}$ can determine at every step during the processing of a tree whether the next input symbol is evaluated in $\mathfrak{A}$ or in $\mathfrak{B}$. Thus we have $T(\mathfrak{C}) = T \cdot_{x_i} S$.

Now we show that $\mathfrak{C}$ is nilpotent. It is trivial that $\bar{a}$ is the trap state of $\mathfrak{A}$ since S is finite. Furthermore, $au = \bar{a}$ implies $au' = \bar{a}$ for every $a \in A$ and $u, u' \in \hat{\Sigma}$ because S is path complete. Moreover, since S is x_i-terminating, every

state $a \in A \setminus \{\bar{a}\}$ with $av = \bar{a}$ implies that $a \in A^{(i)}$ for any $v \in \hat{\Sigma}$. Thus, knowing that T is nilpotent, we easily get that $cw = \bar{b}$ for every state $c \in C$ and path $w \in \hat{\Sigma}^*$ where $|w| \geq k + l$. Therefore, $\mathfrak{C}$ is nilpotent with the nilpotent element $\bar{b}$ and with the degree of nilpotency of not greater than $k + l$. □

Lemma 12. *The DR-language $T_\Sigma(Y)$ is nilpotent for any $Y \subseteq X_n$.*

Proof. Let $T = T_\Sigma(Y)$ be a DR-language for which $Y \subseteq X_n$. We construct the DR ΣX-recognizer $\mathfrak{A} = (\mathcal{A}, a_0, \mathbf{a})$ where $\mathcal{A} = (\{a_0\}, \Sigma)$, $\sigma^{\mathcal{A}}(a_0) = (a_0, \ldots, a_0)$ for all $\sigma \in \Sigma$, and $\mathbf{a} = (A^{(1)}, \ldots, A^{(n)})$ where $A^{(i)} = \{a_0\}$ if $x_i \in Y$, $\emptyset$ otherwise. Obviously, $\mathfrak{A}$ is nilpotent with the nilpotent element a_0 and with the degree of nilpotency of 0. □

A tree language represented by $\eta = \eta_k \cdot_{\xi_k} \ldots \cdot_{\xi_1} \eta_0$ is called a *plain R-chain language*, if $T(\eta_i)$ is finite and path complete, $leaves(T(\eta_i) \setminus X_n) \subseteq \{\xi_{i+1}, \ldots, \xi_k\}$, $root(T(\eta_{i+1})) \cap \Sigma_{T(\eta_i \cdot_{\xi_i} \cdots \cdot_{\xi_1} \eta_0), \xi_{i+1}} = \emptyset$ for all $i \in \{0, \ldots, k-1\}$, and $T(\eta_k) = Z \cdot_{\xi_k} T_\Sigma(Y \cup \{\xi_k\})$ where $Y, Z \subseteq X_n$.

Theorem 3. *Let $T \subseteq T_\Sigma(X_n)$ be a DR-language. T is nilpotent iff it is a plain R-chain language.*

Proof. Let T be a nilpotent DR-language and let $\mathfrak{A}$ be a reduced and normalized nilpotent DR ΣX_n-recognizer that recognizes T. By constructing the plain regular expression $\zeta_{\mathfrak{A}}$ we represent T as a plain R-chain language.

Conversely, let $\eta = \eta_k \cdot_{\xi_k} \ldots \cdot_{\xi_1} \eta_0$ be a plain R-chain language for which $T(\eta) = T$. Using Lemma 12 it is obvious that $T(\eta_k)$ is nilpotent. By repeated use of Lemma 10 and Lemma 11 we get that $T(\eta_{k-1} \cdot_{\xi_{k-1}} \ldots \cdot_{\xi_1} \eta_0)$ is path complete DR-language and is also nilpotent because of Lemma 9. Moreover, we see that $T(\eta_{k-1} \cdot_{\xi_{k-1}} \cdots \cdot_{\xi_1} \eta_0)$ is ξ_k-terminating because every z-path of $T(\eta_{k-1} \cdot_{\xi_{k-1}} \cdots \cdot_{\xi_1} \eta_0)$ is a proper prefix of a ξ_k-path of $T(\eta_{k-1} \cdot_{\xi_{k-1}} \ldots \cdot_{\xi_1} \eta_0)$, $z \in X_n$. Thus using Theorem 2 we get that $T(\eta_k \cdot_{\xi_k} \ldots \cdot_{\xi_1} \eta_0)$ is nilpotent, hence T is nilpotent, too. □

6 Conclusion

We have characterized nilpotent DR-languages by means of plain R-chain languages. To achieve this result, we have stated among others a condition by which the class of DR-languages is closed under the operation of x-product. Unlike in [9] we did not investigate the possibility of reducing plain R-chain languages nor we have investigated the number of auxiliary variables in them, however similar methods seem possible that we have seen in [9].

7 Acknowledgement

The author is thankful to Professor Ferenc Gécseg for his helpful comments and valuable suggestions. The author is also grateful to the anonymous referee whose remarks improved the quality of this paper considerably.

References

[1] Courcelle, B.: A representation of trees by languages I, *Theoretical Computer Science*, **6** (1978), 255-279.

[2] Gécseg, F.: On some classes of tree automata and tree languages, *Annales Academiæ Scientiarum Fennicæ, Mathematica.* **25** (2000), 325-336.

[3] Gécseg, F. and Imreh, B.: On definite and nilpotent DR tree languages, *Journal of Automata, Languages, and Combinatorics.* **9:1** (2004), 55-60.

[4] Gécseg, F. and Imreh, B.: On monotone automata and monotone languages, *Journal of Automata, Languages, and Combinatorics.* **7** (2002), 71-82.

[5] Gécseg, F. and Peák, I.: *Algebraic Theory of Automata*, Akadémiai Kiadó, Budapest 1972.

[6] Gécseg, F. and Steinby, M.: Minimal ascending tree automata, *Acta Cybernetica*, **4** (1978), 37-44.

[7] Gécseg, F. and Steinby, M.: Minimal Recognizers and Syntactic Monoids of DR Tree Languages, in *Words, Semigroups, & Transductions*, World Scientifics (2001), 155-167.

[8] Gécseg, F. and Steinby, M.: *Tree Automata*, Akadémiai Kiadó, Budapest 1984.

[9] Gyurica, Gy.: On monotone languages and their characterization by regular expressions, *Acta Cybernetica*, **18** (2007), 117-134.

[10] Ševrin, L. N.: On some classes of abstract automata. *Uspehi matem. nauk*, **17:6 (108)** (1962), 219.

[11] Virágh, J.: Deterministic ascending tree automata I, *Acta Cybernetica*, **5** (1980), 33-42.

Received 1st January 2008

CONTENTS

ISSN 0324—721 X

Felelős szerkesztő és kiadó: Csirik János